

BENCHMARKING REINFORCEMENT LEARNING FOR SAFE ROBOTICS:  
CONSTRAINTS, ROBUSTNESS AND TRANSFER

by

Zhaocong Yuan

A thesis submitted in conformity with the requirements  
for the degree of Master of Applied Science  
Department of Institute for Aerospace Studies  
University of Toronto

© Copyright 2022 by Zhaocong Yuan

# Benchmarking Reinforcement Learning for Safe Robotics: Constraints, Robustness and Transfer

Zhaocong Yuan

Master of Applied Science

Department of Institute for Aerospace Studies

University of Toronto

2022

## Abstract

Safe learning in robotics aims to deploy robots in real life with complex tasks and safety requirements. To push the agenda of safe learning, a crucial step is establishing common benchmarks that facilitate both reliable evaluations and research development. In this work, we contribute towards this goal by surveying the safe learning literature and proposing a versatile safe learning benchmark suite, **safe-control-gym**. The benchmark implements a variety of safe learning algorithms spanning control to reinforcement learning, it also implements critical features to support safety-relevant evaluations and algorithm development. With **safe-control-gym**, we conduct careful benchmarking on model-free reinforcement learning methods with respect to three metrics of safety: constraint satisfaction, robustness, and transfer performance. We envision **safe-control-gym** to provide a framework that brings various research together, and most importantly to accelerate the progress of safe learning in robotics.

## Acknowledgements

This thesis would not have been possible without the support of many individuals. First and foremost I would like to thank my advisor Angela Schoellig for providing me the opportunity to work on my current thesis topic, and has been supportive throughout my years of graduate study. I would also like to thank Siqi Zhou and Jacopo Panerati for their immense help. Their attention to details, patient guidance and constructive feedback have helped me grow as a researcher. In addition, I am grateful to other members in the safe learning team of Dynamic Systems Lab: Adam Hall, Lukas Brunke and Melissa Greeff, who have been excellent seniors as well as collaborators. Together we have made great contributions to safe learning in RL and control, with several publications and conference workshops. Last but not least, I would like to thank my friends and family, who always believe in me and encourage me to pursue what I want. The challenging journey would have been much tougher without their constant support. I am sincerely grateful to all of them and wish them all the best.

# Contents

|          |                                                    |           |
|----------|----------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                | <b>1</b>  |
| 1.1      | Background . . . . .                               | 2         |
| 1.2      | Contributions . . . . .                            | 3         |
| <b>2</b> | <b>Related Work</b>                                | <b>4</b>  |
| 2.1      | RL Benchmarks . . . . .                            | 4         |
| 2.2      | Safe RL . . . . .                                  | 5         |
| 2.3      | Robust RL . . . . .                                | 6         |
| 2.4      | Transfer Learning . . . . .                        | 8         |
| <b>3</b> | <b>Benchmark Design</b>                            | <b>10</b> |
| 3.1      | Preliminaries . . . . .                            | 10        |
| 3.2      | Core Features . . . . .                            | 11        |
| 3.3      | Software Architecture . . . . .                    | 13        |
| 3.3.1    | Environments and Tasks . . . . .                   | 14        |
| 3.3.2    | Control Baselines . . . . .                        | 17        |
| 3.4      | Initial Benchmark Results . . . . .                | 20        |
| 3.4.1    | Control Performance . . . . .                      | 20        |
| 3.4.2    | Learning Performance and Data Efficiency . . . . . | 22        |
| 3.4.3    | Constraint Satisfaction . . . . .                  | 23        |
| 3.4.4    | Robustness . . . . .                               | 24        |
| 3.5      | Summary . . . . .                                  | 25        |
| <b>4</b> | <b>Benchmark Experiments</b>                       | <b>26</b> |
| 4.1      | Constraint satisfaction . . . . .                  | 27        |
| 4.1.1    | Tasks Description . . . . .                        | 27        |
| 4.1.2    | Baselines . . . . .                                | 28        |
| 4.1.3    | Results . . . . .                                  | 31        |

|          |                                                      |           |
|----------|------------------------------------------------------|-----------|
| 4.1.4    | Ablations . . . . .                                  | 34        |
| 4.2      | Robustness . . . . .                                 | 38        |
| 4.2.1    | Tasks Description . . . . .                          | 38        |
| 4.2.2    | Baselines . . . . .                                  | 39        |
| 4.2.3    | Results . . . . .                                    | 41        |
| 4.2.4    | Ablations . . . . .                                  | 46        |
| 4.3      | Transfer performance . . . . .                       | 50        |
| 4.3.1    | Task Description . . . . .                           | 50        |
| 4.3.2    | Baselines . . . . .                                  | 52        |
| 4.3.3    | Results . . . . .                                    | 54        |
| 4.3.4    | Ablations . . . . .                                  | 60        |
| 4.4      | Summary . . . . .                                    | 63        |
| <b>5</b> | <b>Conclusion</b>                                    | <b>65</b> |
|          | <b>Bibliography</b>                                  | <b>67</b> |
|          | <b>Appendix A 3D Quadrotor Dynamics</b>              | <b>75</b> |
|          | <b>Appendix B Robustness Heatmaps</b>                | <b>77</b> |
|          | <b>Appendix C Augmented Environment Trajectories</b> | <b>80</b> |
|          | <b>Appendix D MMD Similarity Metric</b>              | <b>82</b> |

# List of Tables

|     |                                                                                                                                                                                                                                                                               |    |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 | Feature comparison of <b>safe-control-gym</b> and other safety-focussed reinforcement learning environments, taken from our work [73]. . . .                                                                                                                                  | 11 |
| 4.1 | Ablation on safety critic-based methods with varying safety thresholds and difficulty of constraints in the cartpole task. Highlighted cell shows the best result in the row. . . . .                                                                                         | 35 |
| 4.2 | Ablation on CMDP based methods with varying constraint thresholds in the cartpole task. Highlighted cell shows the best result in the row.                                                                                                                                    | 37 |
| 4.3 | Domain randomization ablation in the cartpole task. . . . .                                                                                                                                                                                                                   | 47 |
| 4.4 | Adversary disturbance scale ablation in the cartpole task. . . . .                                                                                                                                                                                                            | 48 |
| 4.5 | CVaR $_{\alpha}$ threshold ablation in the cartpole task. . . . .                                                                                                                                                                                                             | 49 |
| 4.6 | Final transfer performances (in RMSE) of baselines over the three target environments in cartpole. Highlighted cell shows the best result(s) in the column. . . . .                                                                                                           | 57 |
| 4.7 | Similarity of trajectories between the target environment and source/adapted environments, using a policy trained in the target environment. Highlighted cell shows the best result in the row (excluding the first column). The first column is for self-comparison. . . . . | 59 |
| 4.8 | Ablation on transfer baselines in target-mid environment by varying the amount of source data (in control steps) used for the pretrained policy. Highlighted cell shows the best result in the row. . . . .                                                                   | 61 |
| 4.9 | Ablation on transfer baselines in target-mid environment by varying the amount of target data (in control steps) used for fine-tuning/adaptation. Highlighted cell shows the best result in the row. . . . .                                                                  | 62 |

# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                          |    |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Example benchmarks for machine learning and robotics. <i>Left</i> : KITTI. <i>Mid</i> : DeepMind Control Suite. <i>Right</i> : Safe-control-gym . . . . .                                                                                                                                                                                                                                                                                                | 2  |
| 3.1 | Example configurations in YAML format. . . . .                                                                                                                                                                                                                                                                                                                                                                                                           | 13 |
| 3.2 | Block diagram of <b>safe-control-gym</b> software suite from our work [73].                                                                                                                                                                                                                                                                                                                                                                              | 14 |
| 3.3 | Benchmark environments in PyBullet: <i>Left</i> : CartPole. <i>Right</i> : Quadrotor. . . . .                                                                                                                                                                                                                                                                                                                                                            | 17 |
| 3.4 | Collection of the baseline controllers in <b>safe-control-gym</b> . . . . .                                                                                                                                                                                                                                                                                                                                                                              | 19 |
| 3.5 | Control performance (absolute error w.r.t. reference $\mathbf{x}^{\text{ref}}$ , $ \Delta_e $ ) on the cart-pole stabilization task for different controllers and RL agents. . .                                                                                                                                                                                                                                                                         | 21 |
| 3.6 | Control performance (absolute error w.r.t. reference $\mathbf{x}^{\text{ref}}$ , $ \Delta_e $ ) on the 2D quadrotor tracking task for different controllers and RL agents. . .                                                                                                                                                                                                                                                                           | 21 |
| 3.7 | Comparison of the learning performance (on a logarithmic $x$ -axis) of learning-based control (GP-MPC) and two RL agents (PPO, SAC). Training data is expressed as seconds of simulation time used for collection. Evaluation cost is the RMSE with respect to the target state(s). In cart-pole stabilization, the RMSE uses the full state; in 2D quadrotor tracking, the RMSE only includes the $x$ and $z$ component of the state. . . . .           | 23 |
| 3.8 | In the top plot, fraction of time spent incurring a constraint violation by learning-based control (GP-MPC), vanilla RL (PPO), safety-augmented RL (Safety Layer), and safety-certified control (CBF); in the bottom plot, trajectories for an “impossible” tracking task (with constraints narrower than the reference) for traditional control (Linear MPC), learning-based control (GP-MPC), and safety-augmented RL (PPO with Safety Layer). . . . . | 24 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                        |    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.9  | Robustness of cart-pole stabilization policies learned by traditional (LQR) and learning-based control (GP-MPC) as well as vanilla (PPO) and robust RL (PPO <sup>DR</sup> , RAP) with domain randomization against perturbations in the length of the pole $l$ (left) and a white noise disturbance $\sim \mathcal{N}(0, \sigma_{\mathbf{u}}^2)$ applied to action input $\mathbf{u}$ (right). . . . . | 25 |
| 4.1  | Three aspects of safety for the benchmark: <i>Left</i> : constraint satisfaction. <i>Mid</i> : robustness. <i>Right</i> : transfer performance. . . . .                                                                                                                                                                                                                                                | 27 |
| 4.2  | Learning curves of safe control baselines on the cartpole statbilization task with bounded angle constraint. . . . .                                                                                                                                                                                                                                                                                   | 31 |
| 4.3  | Learning curves of safe control baselines on the quadrotor tracking task with bounded position constraint. . . . .                                                                                                                                                                                                                                                                                     | 32 |
| 4.4  | Final test performances among all baselines and tasks. . . . .                                                                                                                                                                                                                                                                                                                                         | 33 |
| 4.5  | Distribution of safety critic predicted values on trajectories generated from a $\epsilon$ -greedy policy. . . . .                                                                                                                                                                                                                                                                                     | 36 |
| 4.6  | Distribution of the constraint model prediction mean-square-error on trajectories generated from a $\epsilon$ -greedy policy. . . . .                                                                                                                                                                                                                                                                  | 38 |
| 4.7  | Learning curves of the robust baselines on the cartpole stabilization task.                                                                                                                                                                                                                                                                                                                            | 41 |
| 4.8  | Learning curves of the robust baselines on the quadrotor tracking task.                                                                                                                                                                                                                                                                                                                                | 42 |
| 4.9  | Robustness performance benchmark with one varying disturbance in cartpole: <i>Left</i> : pole length (the vertical dash line represents default pole length in training environment). <i>Mid</i> : observation noise (RAAC is not plotted due to bad performance curve lying out of range from the others). <i>Right</i> : external tap force on pole center-of-mass. . . . .                          | 43 |
| 4.10 | Robustness performance benchmark with one varying disturbance in quadrotor: <i>Left</i> : drone mass (the vertical dash line represents default drone mass in training environment). <i>Mid</i> : action noise. <i>Right</i> : external 2D force on drone. . . . .                                                                                                                                     | 43 |
| 4.11 | Robustness performance heatmaps with two varying disturbances (pole length and pole mass with default values 0.5 and 0.1) in cartpole for each baseline. . . . .                                                                                                                                                                                                                                       | 45 |
| 4.12 | Robustness performance heatmaps with two varying disturbances (drone mass and action noise with default values 0.027 and 0) in quadrotor for each baseline. . . . .                                                                                                                                                                                                                                    | 46 |
| 4.13 | Transfer learning performance on cartpole target-easy. . . . .                                                                                                                                                                                                                                                                                                                                         | 54 |
| 4.14 | Transfer learning performance on cartpole target-mid. . . . .                                                                                                                                                                                                                                                                                                                                          | 55 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                        |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.15 | Transfer learning performance on cartpole target-hard. . . . .                                                                                                                                                                                                                                                                                                                                                                         | 55 |
| 4.16 | Example test trajectories with source, target and adapted environments using a policy trained in the target-mid environment. . . . .                                                                                                                                                                                                                                                                                                   | 58 |
| A.1  | Crazyflie diagram in the "X" configuration with motor directions and ordering used in <b>safe-control-gym</b> . . . . .                                                                                                                                                                                                                                                                                                                | 75 |
| B.1  | Robustness performance heatmaps with varying action noise and observation noise in cartpole for each baseline. . . . .                                                                                                                                                                                                                                                                                                                 | 77 |
| B.2  | Robustness performance heatmaps with varying pole length and action noise in cartpole for each baseline. . . . .                                                                                                                                                                                                                                                                                                                       | 78 |
| B.3  | Robustness performance heatmaps with varying drone mass and moment of inertia in quadrotor for each baseline. . . . .                                                                                                                                                                                                                                                                                                                  | 78 |
| B.4  | Robustness performance heatmaps with varying action noise and observation noise in quadrotor for each baseline. . . . .                                                                                                                                                                                                                                                                                                                | 79 |
| C.1  | Example test trajectories with source, target and adapted environments using a policy trained in the target-easy environment. . . . .                                                                                                                                                                                                                                                                                                  | 80 |
| C.2  | Example test trajectories with source, target and adapted environments using a policy trained in the target-hard environment. . . . .                                                                                                                                                                                                                                                                                                  | 81 |
| D.1  | MMD as a distribution distance to measure environment similarities. The three plots correspond to three kinds of perturbed environments based on target-mid cartpole: <i>Left</i> : only pole length changes. <i>Mid</i> : pole length, pole mass and cart mass are all scaled. <i>Right</i> : damping is added to the cart and pole joint. The vertical dash line represents the nonperturbed value in reference environment. . . . . | 83 |

# Acronyms

|                                           |                                                         |
|-------------------------------------------|---------------------------------------------------------|
| <b>MLP</b>                                | Multi-Layer Perceptron                                  |
| <b>RL</b>                                 | Reinforcement Learning                                  |
| <b>DRL</b>                                | Deep Reinforcement Learning                             |
| <b>GP</b>                                 | Gaussian Process                                        |
| <b>MDP</b>                                | Markov Decision Process                                 |
| <b>CMDP</b>                               | Constrained Markov Decision Process                     |
| <b>RWRL</b>                               | Real-world RL suite                                     |
| <b>PPO</b>                                | Proximal Policy Optimization                            |
| <b>SAC</b>                                | Soft Actor Critic                                       |
| <b>LQR</b>                                | Linear Quadratic Regulator                              |
| <b>MPC</b>                                | Model Predictive Control                                |
| <b>SQRL</b>                               | Safety Q-functions for Reinforcement Learning           |
| <b>BVF</b>                                | Backward Value Functions                                |
| <b>DR</b>                                 | Domain Randomization                                    |
| <b>RARL</b>                               | Robust Adversarial Reinforcement Learning               |
| <b>RAP</b>                                | Robustness via Adversary Populations                    |
| <b>CVaR<sub><math>\alpha</math></sub></b> | Conditional Value at Risk at $\alpha$ percentile        |
| <b>WCPG</b>                               | Worst-Case Policy Gradients                             |
| <b>RAAC</b>                               | Risk-Averse Actor-Critic                                |
| <b>GAN</b>                                | Generative Adversarial Network                          |
| <b>GAT</b>                                | Grounded Action Transformation                          |
| <b>GARAT</b>                              | Generative Adversarial Reinforced Action Transformation |
| <b>DARC</b>                               | Domain Adaptation with Rewards from Classifiers         |

# Notation

|                   |                                                    |
|-------------------|----------------------------------------------------|
| $s/x$             | observation/state                                  |
| $a/u$             | action/input                                       |
| $P(s, a)/f(s, a)$ | transition function/system dynamics                |
| $r(s, a)$         | reward function                                    |
| $\mu(s)$          | initial state distribution                         |
| $\gamma$          | MDP discount factor                                |
| $\pi$             | policy                                             |
| $V(s)$            | state conditional value function                   |
| $Q(s, a)$         | state-action conditional value function/Q function |
| $c(s, a)$         | CMDP constraint function                           |
| $d$               | CMDP total constraint threshold                    |
| $\lambda$         | Lagrange multiplier                                |
| $\epsilon$        | safety critic threshold                            |
| $\alpha$          | CVaR percentile threshold                          |

# Chapter 1

## Introduction

Reinforcement Learning (RL) in robotics has achieved remarkable success in recent years, pushing the boundaries of complex control tasks and bringing their real-life applications up on the agenda. Despite the prosperity in development, concerns are raised regarding the reusability of various RL works which are often designed for specific environments or objectives. To address such concerns, an important approach is setting up a common benchmark to facilitate cross-algorithm evaluations. Existing literature has shown many examples of success such as the KITTI Vision Benchmark [29] and DeepMind MuJoCo Control Benchmark [65]. There also has been a running effort to develop high-quality datasets and benchmarks for machine learning research, bringing broader impact even to conference venues [67].

On the other hand, safety is essential for robot operations. Recent works in RL has upweighted the importance of ensuring safety in their learning frameworks [28, 5, 6]. Safety requirements also differ based on the target scenarios. Among them, the most straight-forward metric is constraint satisfaction, which has wide popularity and has motivated plenty of works in safe RL. Robustness is another common metric from a broader sense of safety, which focuses on maintaining optimal performance despite the presence of disturbances. Yet another related metric is transfer performance, which evaluates a policy against more drastic environment variations. The works of Sim2Real [64, 10] is representative in transfer learning for RL, also bearing great potential in pushing RL to real-world robotics.

In this work, we explore the joint topic of benchmarking and safe learning for RL in the domain of robotics. In contrast to other works that push for the state-of-the-art (SOTA) performance in a particular setting, we focus on building up tools to facilitate

safe learning, designing common comparisons of control algorithms, and inspecting their quantitative differences in the aforementioned safety metrics: constraint satisfaction, robustness and transfer performance. To cover a wider range of target scenarios but without the expensive cost in design iterations, this work makes use of simulation environments in PyBullet [16] for both training and evaluation. Lastly, this work is compiled in an open-source software stack named **safe-control-gym**<sup>1</sup>, which includes the collection of implementations, experiments and other resources to facilitate future research in safe-learning benchmark in robotics.

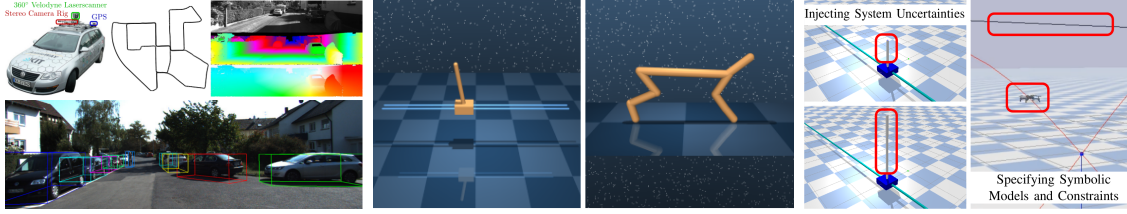


Figure 1.1: Example benchmarks for machine learning and robotics. *Left*: KITTI. *Mid*: DeepMind Control Suite. *Right*: Safe-control-gym

## 1.1 Background

Safe learning in robotics is at the intersection of multiple lines of research, and benchmarking them is non-trivial due to the differences in algorithm assumptions and targeted scenarios. An example is the accessibility of system information, where control-oriented approaches such as linear-quadratic regulator (LQR) and model predictive control (MPC) often assume prior models of system dynamics and more data-driven approaches like model-free RL only rely on input-output data from the system. This poses a challenge for the benchmark design which should ideally provide the support to these different approaches. To address the challenge, we conduct an extensive literature review and contribute towards a recent safe-learning review for robotics [8], from which the benchmark is based on in terms of the common language for problem formulation and the choice of representative baselines for comparison.

Furthermore, the proposed benchmark or **safe-control-gym**, is inspired from several other RL benchmark projects covered in Section 2.1. Specifically, the work of Real-World RL Suite (RWRL) [22, 21] is the closest to ours, which aims to identify key challenges in RL for real-world use and performs some initial benchmarking. Our work partially overlaps with RWRL in evaluation metrics, but greatly extends both the

<sup>1</sup>URL link: <https://github.com/utiasDSL/safe-control-gym>

testing environments and control baselines for benchmarking. In addition, we adopt the PyBullet physics engine instead of MuJoCo in RWRL, which has the advantage of being fully open-source (at the time of development<sup>2</sup>) and having wider customized usage. A description of **safe-control-gym** as well as the benchmarking results it facilitates are compiled in another recent work of ours [73].

## 1.2 Contributions

This work focuses on benchmarking RL algorithms in the context of safe learning in robotics, with contributions summarized as follows:

- a novel benchmark suite for safe learning control, with environments and utilities to support development of safety-aware algorithms.
- baseline implementations of representative control approaches, including standard control, safe learning-based control, safe RL, robust RL and RL with transfer learning.
- benchmark results and analysis with respect to typical safety metrics: constraint satisfaction, robustness and transfer performance.

To highlight, our effort in safe learning for robotics has led to two published works: the safe learning survey [8] in Annual Reviews and the safe-control-gym paper [73] in IROS/RAL. The thesis is based on these publications but focuses on the selected personal contributions which include the overall design of the software suite, development, and benchmarking of RL agents in various safety contexts.

To further clarify the connection between the aforementioned publications and this thesis, Chapter 2 expands on Section 3.2 of the survey [8] about safety in RL. Chapter 3 builds upon the safe-control-gym paper [73] but further elaborates on the benchmark design and safety comparisons between control and RL approaches.

Lastly, Chapter 4 covers unpublished work that includes newer, more in-depth benchmarking results on a set of model-free approaches from safe RL, robust RL and transfer learning. Detailed analyses are also performed on selected classes of methods to guide future practices on RL for safe robotics.

---

<sup>2</sup>MuJoCo also becomes fully open-source as of May, 2022.

# Chapter 2

## Related Work

Reinforcement learning (RL) is a learning framework for decision making. In RL, a task is defined as a Markov Decision Process (MDP) [4] denoted  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mu, \gamma)$ , where the elements represent state or observation space, action space, transition dynamics, reward function, initial state distribution and discount factor. RL learns the optimal policy as a mapping  $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$  with respect to the expected return  $J(\pi) = \mathbb{E}_{\pi}[\sum_i \gamma^i r(s_i, a_i)]$ . Typical RL methods fall under several categories: value-based, policy gradient and actor-critic. Deep RL uses neural networks to further augment these methods for more flexible parameterization and learning. In the following, we cover the related works to benchmark design in RL and safe control in RL, a more general and comprehensive description of safe-learning control can be found in our published review [8].

### 2.1 RL Benchmarks

Benchmarks are designed for a particular problem formulation of interest and provide common evaluation protocols for similar algorithms, often serving as the cornerstone that greatly advances related research. Well-known examples such as ImageNet [18] in computer vision and KITTI [29] in autonomous driving have lead the momentum in developing high-quality benchmarks and datasets for machine learning. The RL and control community have also seen a rise of benchmark projects that specialize in different directions of policy learning, for example, RL benchmark for continuous control [20], model-based RL benchmark [70], unsupervised RL benchmark [39], and offline RL benchmark [32]. Among them, the safe RL benchmarks [40, 54, 22, 21] are the most related to our work but not without distinctions. Concretely, the work [40] uses grid-world environments which remain too simple for robotic tasks.

Safety Gym [54] mostly focuses on safe exploration but only on navigation tasks, and only tests with a narrow set of baselines. Real-world RL Suite (RWRL) [22, 21] builds upon the MuJoCo benchmark and has a diversity of testing scenarios, ranging from different tasks to different safety settings; yet the tested baselines are safety-agnostic which are less informative in safety comparison and harder to reuse for extension. Compared to RWRL, our work further includes other common robotics tasks and provides a set of representative control baselines that target different safety requirements, making not only flexible benchmark environments but also reusable algorithm implementations for related research. Notably, our work is compatible with other RL baseline projects such as garage [15], RLlib [41] and Stable-Baselines3 [52], which can be used as alternatives to develop the equivalent safe control baselines.

Furthermore, proper metrics are essential for benchmark evaluations since they reflect prior knowledge in the problem and encode the desired behaviors of good control policies. Common RL methods are often evaluated with the expected return, which could be limited, for example, in safety-critical tasks that require high confidence of good performance. To mitigate this, recent works have proposed diverse metrics for RL performance, aiming to build a more comprehensive profile. The work [48] introduces bsuite which evaluates RL agents based on robustness to noisy rewards, exploration efficiency, scalability and more. The work [9] proposes risk-aware performance metrics based on inter-quartile range and conditional value at risk of the returns. RWRL [22, 21] also includes multiple metrics designed for practical RL, such as inference speed, explainability and robustness to various noises. Since our work targets safe-learning which has various requirements, naturally we also use different metrics — constraint satisfaction, robustness and transfer performance, to evaluate control policies. But since not all policies are trained to maximize all the metrics, we only benchmark those that fit in the corresponding category.

## 2.2 Safe RL

RL with constraints is often modeled as a Constrained Markov Decision Process (CMDP) [2] denoted by  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mu, \gamma, c, d)$ , where  $c$  is the constraint function and  $d$  is the threshold to the total constraint cost  $J_c(\pi) = \mathbb{E}_\pi[\sum_i \gamma^i c(s_i, a_i)]$ , both  $c$ ,  $d$  can be vector-valued to represent multiple constraints. CMDP then solves a constrained optimization problem  $\max_\pi J(\pi)$  s.t.  $J_c(\pi) \leq d$ . To tackle this problem, a direct approach is to apply the Lagrangian method and transform into unconstrained

optimization [11, 42]. However, a key issue in safe RL is to avoid or reduce constraint violations during learning, and the Lagrangian method cannot guarantee it. Furthermore, typical RL methods in continuous control rely on sampling-based updates. In order to ensure safety during each learning update, the constraint cost will raise an off-policy evaluation problem which is known to be challenging.

Subsequent safe RL works address these problems in different ways. In CPO [1], the authors leverage surrogates to the expected return and constraint cost to derive trackable policy updates that ensure monotonic policy improvement and near constraint satisfaction. In works [12, 13], Lyapunov functions are used to transform the total constraint cost into step-wise constraints that allow more efficient evaluation. But a baseline policy that satisfies the constraints are required. In BVF [56], the total constraint cost is decomposed into a forward and a backward value function, which are learned jointly with typical Temporal Difference (TD) algorithms and used to impose the total constraint cost constraint. These methods all share the common technique of approximating the total constraint cost with a constraint value function similar to the value functions  $V(s)$ ,  $Q(s, a)$  in RL. Hence they allow efficient estimation given only single-step information which alleviates the off-policy evaluation problem, and at the same time ensures approximately safe policy updates by enforcing the learned constraint value function to stay within given threshold.

Another popular approach to enforce constraint satisfaction is to use safety filters. A typical filter works by transforming output from a potentially unsafe policy to a safe action with the least modification. An example in control is using control barrier functions (CBF) [61] in the filter, and in RL using a learned, linearized constraint function in the filter as in the Safety Layer [17]. In addition, the filter can also incorporate learned constraint value function, or safety critic  $Q_c^\pi(s, a)$  as called in SQRL [59], Recovery RL [62] and CSC [6]. The basic form of the filter is to accept a proposed action  $a \sim \pi(\cdot|s)$  if it is constraint satisfying  $Q_c^\pi(s, a) \leq d$ , otherwise resort to a fallback action which is produced by different schemes across these works.

## 2.3 Robust RL

Safety in RL can also be encouraged by enforcing robustness from different perspectives. One view of robustness is against performance variance due to the intrinsic stochasticity of the policy or the environment. Viewing the RL return as a random variable, this modifies the goal of RL from maximizing the mean value to another

measure that should capture both mean and variability of the return distribution. The resulting policy will then concentrate on region of high returns. Another view pursues robustness with respect to worst-case performance, typically with noise or disturbances in the system. This can be formulated as a min-max optimization problem  $\max_{\pi} \min_{f \in \mathcal{F}} J(\pi, f)$  where  $f$  is the disturbed dynamics from a given class  $\mathcal{F}$  to be robust against. Other views of robustness include learning stability, sensitivity to parameters and generalization within similar tasks. In short, a robust RL or risk-aware RL agent is often based on one or more of these perspectives. For example, in CARL [74] and SAVED [63], a model-based RL algorithm using probabilistic ensemble model [14] is adapted to estimate the probability of collision during planning, hence reducing the risk of unsafe behaviors. In WCPG [60] and ORAAC [66], distributional RL is used to represent the return distribution. They then optimize with a risk measure called Conditional Value-at-Risk (CVaR) [11] that captures the lower percentile of returns, which is equivalent to improving worst-case performance.

A specific formulation of robust RL against worst-case scenario is adversarial RL. Similar to a Generative Adversarial Network (GAN) [30], there is a protagonist and an adversary agent. In adversarial RL, the protagonist learns to maximize the expected return while the adversary learns to inject harder disturbances to the system, the two agents are updated alternatively. RARL [51] is an instantiation of such approach, and models the adversary as disturbance forces on the robot. The work [49] uses value function ensembles to define a risk-aware objective, which the protagonist and adversary seek to optimize in opposite directions. Another later work [68] extends RARL by using multiple adversaries to prevent the protagonist from overfitting to a single mode of disturbance.

Domain randomization (DR) is also a popular method for achieving robustness in RL. The idea is to train the RL agent under randomized variations of a nominal system such as changing robot inertial properties, friction coefficients or visual scenes, hence forcing the agent to have good generalization. This approach can operate with any off-the-shelf RL algorithm and greatly benefit from the recent development of parallel computing. But some notable downsides include domain-specific knowledge to design the system randomization, large amount of training resources and lots of manual tuning. In addition, DR might result in conservative policies that compromise task performance in exchange for generalization, which is not always desired. Despite these, DR has shown many success in practice, ranging from quadrotor flight [55, 43] to manipulator tasks [64, 10]. Other works [53, 46] replace the uniform randomization

strategy with more informative, guided randomization that restricts the search space to speed up learning and reduce conservatism.

## 2.4 Transfer Learning

Transfer learning in RL aims to learn a policy for a target domain by training or reusing data from one or more source domains. It is often assumed that the target domain is expensive or difficult for data collection such as real-life robotic platforms, while the source domain is easily modifiable and can generate excessive training data, as in the case of simulators. In RL for robotics, simulators are a powerful tool to accelerate learning, but they are not perfect in modeling real-life dynamics. This reality gap [47] must be considered and a common practice with transfer learning is to pre-train a policy extensively in simulators, then fine-tune on the actual robot with a limited budget. Such process can go through further iterations to slowly improve the policy and reduce the reality gap. If one aims to push RL for real-life robots, transfer learning will be a necessary component and hence, benchmarking transfer learning techniques will also be an integral part of safe learning in RL. Specifically, we focus on domain adaptation, where the RL agent mainly learns from source data but can also query a limited amount of target data to adapt its behavior.

Given some target data and a simulator used for training, a natural idea is to adapt the simulator so it mimics the target dynamics as close as possible. System identification can help in finding the best simulator parameters that match the target data, but it is limited when the reality-gap is beyond parameter mismatch. To fix this, a common approach in transfer learning is to augment the simulator with neural networks, such that they learn a more flexible class of model mismatch. Using this idea, SimGAN [38] learns state-action conditional functions of the parameters. GAT [35] and GARAT [19] learn an action transformation on top of the simulator, such that the combined dynamics matches the real one. Furthermore, adapting the simulator requires a similarity measure between source and target data, which will serve as either a regression loss or RL reward for learning. GAT optimizes for maximum likelihood of the target data conditioned on the simulator as the similarity measure. SimGAN and GARAT instead adopt a distribution matching approach, implemented as a discriminator that are optimized to distinguish between source and target data. MATL [72] and DARC [23] also leverage distribution matching but use the similarity measure directly as a reward shaping term, avoiding the need to adapt the simulator.

Besides, domain randomization can also make use of the similarity measure given access to target data. SimOpt [10] and ADR [46] leverage this idea to effectively produce better randomizations for learning a robust policy. Although DR does not adapt to a particular target environment, the learned policy could still generalize to the target given sufficient similarity to the randomized systems it has trained on. So DR is also a common baseline for benchmarking transfer performance.

# Chapter 3

## Benchmark Design

This chapter introduces **safe-control-gym** as the primary benchmark suite for safe-learning in robotics and is based on our paper [73]. Among the aforementioned related works in Section 2.1, we have not found one that completely satisfy our requirements for implementing and benchmarking the various safe-learning approaches. RWRL [22, 21] is the closest to this purpose, but it also has several limitations considering: (i) the lack of utility support such as prior models and symbolic optimization for control-oriented approaches; (ii) the given robots and tasks are mainly suitable for simulation, lacking support to real-life robotics. Hence, we are motivated to develop a new benchmark that better cover our needs.

### 3.1 Preliminaries

A widely used standard for RL environment design is the OpenAI Gym interface [7], which defines a basic framework on the agent-environment loop helping to decouple implementations. A gym environment includes the common elements:

- **observation\_space**, **action\_space** define the types and ranges of state or observation from the environment output and action from the RL agent output.
- **reset()** implements initialization of the environment.
- **step()** implements forward dynamics of the environment given action input.
- **render()** implements an optional display of the current environment.

To facilitate high-fidelity simulation of robot dynamics, **safe-control-gym** uses PyBullet [16] to build the learning environments. URDF files are used for robot specifications, which are easily reusable with ROS [45] for real-world experiments.

## 3.2 Core Features

To support development of various safe learning approaches as mentioned in Section 2.2, 2.3, 2.4, **safe-control-gym** implements a set of features augmenting the basic Gym environments. Although this work focuses on comparing RL methods, the benchmark is applicable to traditional control and safe-learning-based control as well. These methods have long been tackling the problem of safety and have mature usage in practical robotic systems, unlike RL that often falls short due to the high data demand and lack of guarantees. Hence, the design of **safe-control-gym** also takes into account of the control-oriented approaches, and subsequent comparisons also include them as they are valuable baselines.

Table 3.1: Feature comparison of **safe-control-gym** and other safety-focussed reinforcement learning environments, taken from our work [73].

|                                   | Physics Engine | Rendering Engine     | Robots                    | Tasks                       | Uncertain Conditions           | Constraints  | Disturbances           | Gym API | Symbolic API |
|-----------------------------------|----------------|----------------------|---------------------------|-----------------------------|--------------------------------|--------------|------------------------|---------|--------------|
| <b>safe-control-gym</b>           | Bullet         | TinyRenderer, OpenGL | Cart-pole, Quadrotor      | Stabilization, Traj. Track. | Inertial Param., Initial State | State, Input | State, Input, Dynamics | Yes     | Yes          |
| <b>ai-safety-gridworlds</b> [40]  | n/a            | Terminal             | n/a                       | Grid Navigation             | Initial State, Reward          | State        | Dynamics, Adversaries  | No      | No           |
| <b>safety-gym</b> [54]            | MuJoCo         | OpenGL               | Point, Car, Quadruped     | Goal, Push, Button          | Initial State                  | State        | Adversaries            | Yes     | No           |
| <b>realworldrl-suite</b> [22, 21] | MuJoCo         | OpenGL               | Humanoid, Cart-pole, etc. | Swing-up, Walk, etc.        | Inertial Param., Initial State | State        | State, Input, Reward   | No      | No           |

**Symbolic Models:** We augment each environment with prior models on the system dynamics, cost function and state-action constraints implemented with CasADi [3], an open-source symbolic framework with a rich toolbox of nonlinear optimization techniques. The symbolic priors, along with other crucial information such as initial state and task references are returned upon environment reset. We further pre-compute some common functions of interest for downstream implementations such as dynamics integration (both discrete and continuous), dynamics linearization, cost gradient and hessian. With these, it is then easy to construct a safe control objective and leverage existing optimizers as done in many control approaches.

**Constraints:** Each environment is also augmented with the ability to specify and evaluate constraints  $c(s, a)$ . Specifically, the environment observation space and action space are used to define default state and input constraints. Additional constraints can be defined in environment configuration. Each constraint is implemented

with a slack parameter for constraint relaxation or tightening, a symbolic model for CasADi optimization, and various function templates such as linear, bounded and quadratic. Notably, the same constraint specification can be used in the control algorithm where custom constraints are needed for optimization. At each environment step, the current constraint values and any possible constraint violation are returned as auxiliary information for the next policy control step, the violation signal is also used to determine episode termination, and the constraint values are used for reward shaping if enabled. The overall constraint API should ideally capture most usage in both control-oriented approaches as well as safe RL approaches.

**Disturbances:** Various types of disturbances can be applied to the environment. Although it is difficult to capture all the differences between a simulation environment and its corresponding real-world setup, we include a representative set of disturbances that appear most often in our benchmark baselines: initial state randomization, system parameter randomization, observation noise, action noise and external forces. Specifically, the randomizations are implemented with configurable distributions in environment arguments, providing the support for domain randomization approaches as well as benchmarking for transfer learning methods. The noise disturbances also come with configurable function templates to support gaussian noise, uniform noise, step noise and impulse noise. The external force disturbance is implemented via PyBullet API and is environment-specific. Moreover, the noise and external force disturbances can be set outside of the environment, which is designed for adversarial training. The overall disturbance API also serves to enable implementation and comparison of robust control and robust RL approaches.

**Configuration:** To configure the aforementioned safety options of the environment, **safe-control-gym** uses a YAML-based configuration system that manages all the experiment parameters for both the learning agent and the environment. The system merges several sources of parameters per experiment, including default parameters for the agent or environment, overriding parameters as additional YAML files and overriding parameters as command line arguments. This provides great flexibility in fast iteration as well as support for reproducibility, while not restricting to a rigid format. Example configuration files can be seen in Figure 3.1.

```

task_config:
  cost: quadratic # The reward function.
  ctrl_freq: 50
# The control input frequency.
  pyb_freq: 1000
# PyBullet's stepping freq.
  constraints:
# Constraints on the system.
    - constraint_form: bounded_constraint
      lower_bounds: [-1, -0.2, -0.3, -0.05]
      upper_bounds: [1, 0.2, 0.3, 0.05]
      constrained_variable: STATE
  disturbances:
# Disturbances on the system.
  observation:
    - disturbance_func: white_noise
      std: 0.05

algo_config:
# model args
  hidden_dim: 64
# runner args
  max_env_steps: 1000000
  warm_up_steps: 1000
  rollout_batch_size: 4

task_config:
  ctrl_freq: 50
  pyb_freq: 1000
  episode_len_sec: 5
  normalized_rl_action_space: True
  info_in_reset: True
# task
  task: stabilization
  cost: rl_reward

```

Figure 3.1: Example configurations in YAML format.

### 3.3 Software Architecture

Along with the core features, **safe-control-gym** provides implementation of a set of benchmark environments and baseline controllers. The environments subclass from the Gym API and are configured to use the safe learning utilities described in Section 3.2. The controllers define the learning loop given collected data. Since the modules for environments and controllers are decoupled, other external packages for RL baselines such as garage [15] and Stable-Baselines3 [52] can be used as alternatives. Data collection is done by the agent-environment interaction loop as in common RL. But the control agent is provided with additional information during initial reset and each subsequent step to facilitate safe decision making.

Moreover, we explicitly distinguish between the control frequency ( $f_{\text{ctrl}}$ ) and simulation frequency ( $f_{\text{PyB}}$ ) by allowing them to be configured in environment arguments. Each control step can correspond to multiple simulation steps taken by the physics engine, which helps to tune the discretization of dynamics and also helps to trade-off agent inference time. In case of MPC-based algorithms, planning at each step often involves expensive look-ahead operations of the dynamics and cost, but can be alleviated by lowering the control frequency. This also raises an issue for fair benchmarking given baseline controllers operating in different frequencies. To address this in experiments, we fix the simulation frequency of environments and scale the per-step rewards accordingly for each baseline to account for different control frequencies. The same operation is applied to other safety metrics such as episodic constraint violations.

Robot and scene description are defined using URDF files, which are included as

environment assets. The adversarial agent sets disturbances from outside the environment, and can be implemented as a separate controller with its own learning loop. Figure 3.2 provides an overview of the software architecture and the aforementioned design details.

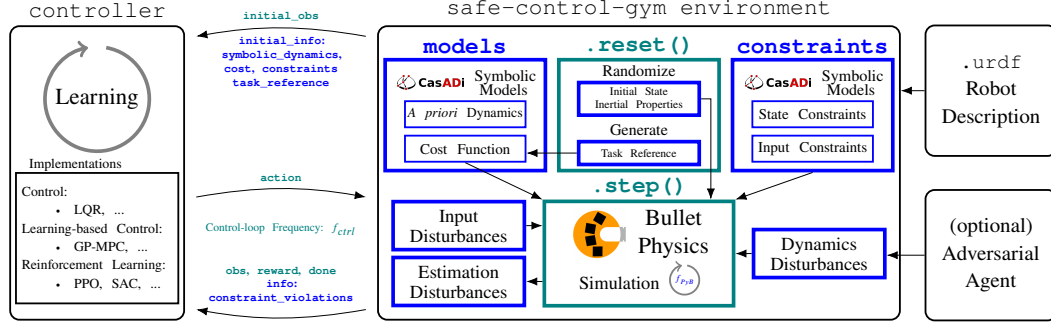


Figure 3.2: Block diagram of **safe-control-gym** software suite from our work [73].

### 3.3.1 Environments and Tasks

The current **safe-control-gym** contains a CartPole environment and a Quadrotor environment. Both are commonly used environments in RL and control, with the advantages of **i)** they allow simple torque-based control from observed states, which helps in benchmarking the variable factors in safety performance compared to complex domains such as high-dimensional or vision-based control; **ii)** they come with tractable dynamics model that can be used as priors, which play an important role in facilitating safety and allow comparison of both RL approaches and traditional control; **iii)** they support different control tasks which are also practical for real-life setup. This enables the benchmark to connect with real robots which is useful for converting the benchmark results to deployment.

**CartPole:** The system consists of an actuable cart on a rail, with a pole attached via a prismatic joint. The cart has mass  $m_c$ , the pole has mass  $m_p$  and length  $2l$ . The gravitational constant is  $g$ . The state of system is  $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$ , where  $x$  is position of cart along the rail and  $\theta$  is angle of pole with respect to the vertical axis, the dotted quantities are velocities. The action input of the system is the 1D force  $\mathbf{u} = F$  on the cart's center-of-mass (CoM). The equation of motion is given in [26] as

$$\ddot{x} = \frac{F + m_p l (\dot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta)}{m_c + m_p} \quad (3.1a)$$

$$\ddot{\theta} = \frac{g \sin \theta + \cos \theta \left( \frac{-F - m_p l \dot{\theta}^2 \sin \theta}{m_c + m_p} \right)}{l \left( \frac{4}{3} - \frac{m_p \cos^2 \theta}{m_c + m_p} \right)} \quad (3.1b)$$

CartPole domain implements the stabilization task and rewards a control agent by keeping the pole upright at the origin. To accommodate both control and RL approaches, we use the negative quadratic cost as reward but allow algorithm-specific reward shaping, for example, control approaches often tune the quadratic weights  $\mathbf{Q}, \mathbf{R}$  and RL approaches often apply normalization on rewards.

$$r = \mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i + \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i \quad (3.2a)$$

For interfacing the control agent, CartPole uses direct state as observation and continuous action, which is preprocessed (with clipping and rescaling) before being fed to the simulation. Depending on where the pole is initialized, CartPole stabilization can be further divided into two tasks, the normal stabilization task with  $|\theta_0| < \pi/2$  and the harder swing-up task otherwise.

**Quadrotor:** The Quadrotor domain is based on the **gym-pybullet-drones** project [50], and includes several environments with increasing complexity. Each environment uses the crazyflie drone model with mass  $m$ , moment of inertia  $I$  and arm length  $l$ . The drone is commanded via direct thrusts on the four propellers  $T_i$ ,  $i \in \{1, 2, 3, 4\}$ . The 1D quadrotor restricts the drone movement to the vertical z-axis, with state  $\mathbf{x} = [z, \dot{z}]^T$  and action as the total thrust  $\mathbf{u} = T = \sum_i T_i$ . The dynamics is then

$$\ddot{z} = T/m - g \quad (3.3a)$$

The 2D quadrotor constrains the drone on the x-z plane, with state  $\mathbf{x} = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^T$  where  $\theta$  is the pitch angle. The action is the thrust pair on each side of the drone's body frame y-axis  $\mathbf{u} = [T_{14}, T_{23}]^T = [T_1 + T_4, T_2 + T_3]^T$ . Given  $d = l/\sqrt{2}$  as the

effective moment arm, the 2D quadrotor dynamics is

$$\ddot{x} = \sin \theta (T_{14} + T_{23}) / m \quad (3.4a)$$

$$\ddot{z} = \cos \theta (T_{14} + T_{23}) / m - g \quad (3.4b)$$

$$\ddot{\theta} = (T_{14} - T_{23}) d / I_{yy} \quad (3.4c)$$

The 3D quadrotor allows full flexibility of the drone movement, with state containing the drone position, linear velocities, orientation (as Euler angles) and angular velocities in the body frame  $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]^T$ . The action is the four motor thrust forces  $\mathbf{u} = [T_1, T_2, T_3, T_4]^T$ . The full dynamics of 3D quadrotor can be found in Appendix A.

All the quadrotor environments support stabilization and tracking tasks. For stabilization, the drone is commanded to approach and hover at a given reference state. For tracking, the drone needs to follow a sequence of reference states. Similarly, we use the negative quadratic cost with respect to the reference states  $\mathbf{x}_i^{\text{ref}}$  and actions  $\mathbf{u}_i^{\text{ref}}$  as the basic reward. The reference states and actions come from a trajectory generator and they are pre-defined upon episode start.

$$r = (\mathbf{x}_i - \mathbf{x}_i^{\text{ref}})^T \mathbf{Q} (\mathbf{x}_i - \mathbf{x}_i^{\text{ref}}) + (\mathbf{u}_i - \mathbf{u}_i^{\text{ref}})^T \mathbf{R} (\mathbf{u}_i - \mathbf{u}_i^{\text{ref}}) \quad (3.5a)$$

The environments also expose direct state as observation and apply preprocessing to the input action. Specifically for RL approaches, we either concatenate the current reference to the state in observation, or use the difference between them; in tracking task we also enable adding future references or differences to the observation to facilitate better planning. Besides, actions are denormalized to center around the hovering thrust during preprocessing which has been shown to improve learning.

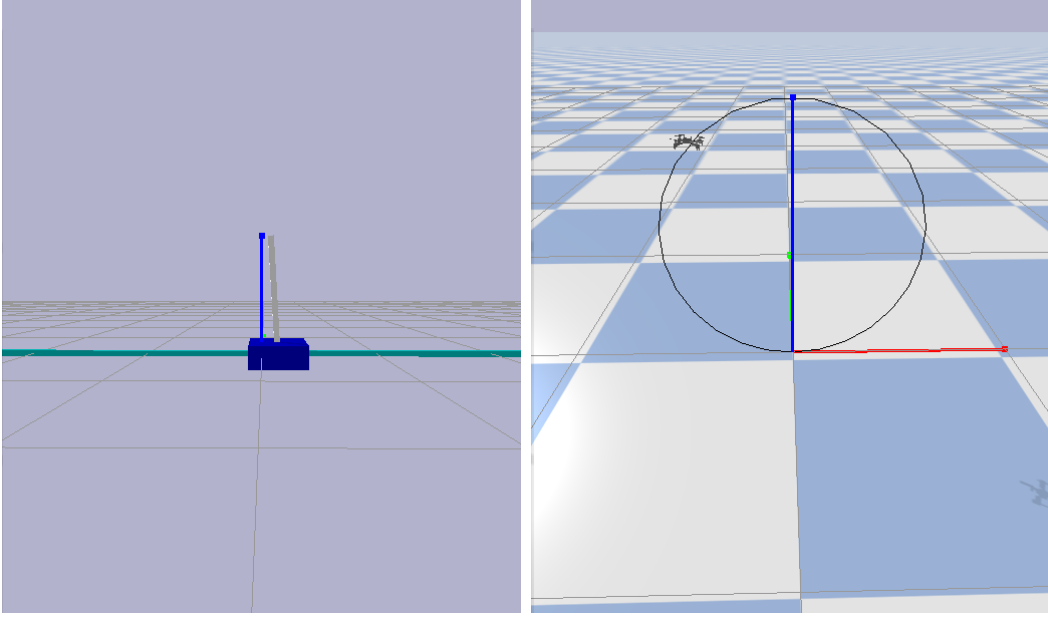


Figure 3.3: Benchmark environments in PyBullet: *Left*: CartPole. *Right*: Quadrotor.

### 3.3.2 Control Baselines

Guided by the safe-learning review [8], **safe-control-gym** implements some of the representative safe-learning approaches. They leverage different utilities from the environments and cover different aspects of safety in learning, which should ideally serve as useful baselines for benchmark comparison and future work.

**RL baselines:** We include PPO [57] and SAC [33, 34] as safety-unaware baselines. The two algorithms are commonly used in other robotics works as they are simple, effective and robust for learning. They are also representative in on-policy and off-policy setting respectively, which are the two main categories of RL. Since both PPO and SAC are model-free, we additionally include PETS [14] as a model-based RL baseline. These methods are not only candidates for benchmark comparisons, but also the backbone controllers as we develop other safe-learning approaches.

**Safe RL:** We include a simple safe RL baseline by applying reward shaping to PPO. Specifically upon constraint violation, a penalty is added to the reward at each step. We also implement the Safety Layer work [17] with PPO as the underlying agent. Safety Layer deals with state constraint and learns the linear approximation of the constraint at the next state with respect to the current state and action during pre-

training. Then it uses the approximated constraint to filter the PPO output, transforming a potentially unsafe action that might violate constraint at the next state to a safe action with minimal deviation. Works of [12, 13, 56] transform the CMDP trajectory constraint into step-based constraint by learning constraint value function, then a similar action filtering can be applied given there is access to the linearization of the transformed constraint. We include these ideas building on SAC. Another set of related baselines also learn the constraint value function or safety critic but filter unsafe actions in different ways. We include two recent examples: SQRL and Recovery RL. In SQRL, action filtering is done via rejection sampling, avoiding any action with constraint value that exceeds the safety threshold. In Recovery RL, a safe policy is learned separately and will be resorted to when an unsafe action is proposed. In addition, the safety critic can also be used in the Lagrangian method as another simple safe RL baseline.

**Robust RL:** A common baseline for robust RL is domain randomization, which can be realized with the disturbance API in **safe-control-gym**. Aside, we include two adversarial RL baselines: RARL [51] and RAP [68], we follow the settings in the original works where RARL uses adversarial disturbance force and RAP uses adversarial action noise. Both the protagonist and adversary agent use PPO for learning. Risk-aware learning is also important in robust RL. We include recent works that are based on distributional RL and the CVaR risk measure: WCPG [60] and ORAAC [66]. Since we do not focus on the offline learning setting, we use the online version of ORAAC or simply RAAC.

**Transfer Learning:** To benchmark transfer performance, we focus on approaches that tackle domain adaptation. In this setting, the agent is trained with a source domain and limited data from the target domain, then evaluated in the target domain. The ability to adapt to a similar but new environment is important for safe robot operation, since an increasing number of the RL controllers are trained in simulation given its benefits in free training data and scalability by parallel compute. But simulation still cannot model every detail of the real-life dynamics, making RL controllers brittle to changes between the training and target environments. A common practice to address this problem is system identification, which we include as a baseline. We also include domain randomization as it has been proved to be a decent baseline in the transfer setting. For more advanced techniques, we look into transfer via distribution matching. We adapt SimGAN [38] as a baseline that makes use of simulator as prior model, and learns to augment the simulator with highly flexible

neural-network-based functions. We also include GAT [35] and GARAT [19] which learn an action transformation on top of the simulator, that serves to produce similar dynamics from the target domain. Furthermore, we add DARC [23] to the list. Unlike previous two transfer baselines, DARC does not try to augment the simulator, but instead use reward shaping that encourages agent learning to favor good transfer performance. Notably, SimGAN, GARAT and DARC all leverage a discriminator to facilitate distribution matching. This is done via adversarial learning, where the discriminator is optimized to distinguish between source data and target data; while the agent is optimized to produce similar data in either domains.

**Control-oriented:** In addition to the many RL baselines, we also include approaches from traditional control such as LQR, Model-Predictive-Control (MPC), and some recent works in safe-learning-based control where applicable for comparison. For example, GP-MPC [37] learns a Gaussian Process on top of the prior dynamics model, thus overcoming the initial imperfect knowledge. It can be used to benchmark both constraint satisfaction and robustness; if the GP is allowed to fine-tune given target data, it is applicable in transfer settings as well. Besides GP-MPC, learning-based Control Barrier Functions (CBF) [61] is another related baseline for constraint satisfaction that does action filtering similar to Safety Layer. Again, we use an underlying PPO agent that can output unsafe actions, and apply the learned CBF to produce safe actions.

A summary of the aforementioned baseline agents is shown in Figure 3.4. Although the list is currently limited, there is an active effort to include more approaches in **safe-control-gym** and to push for other safe-learning scenarios, such as those suggested in the review [8].

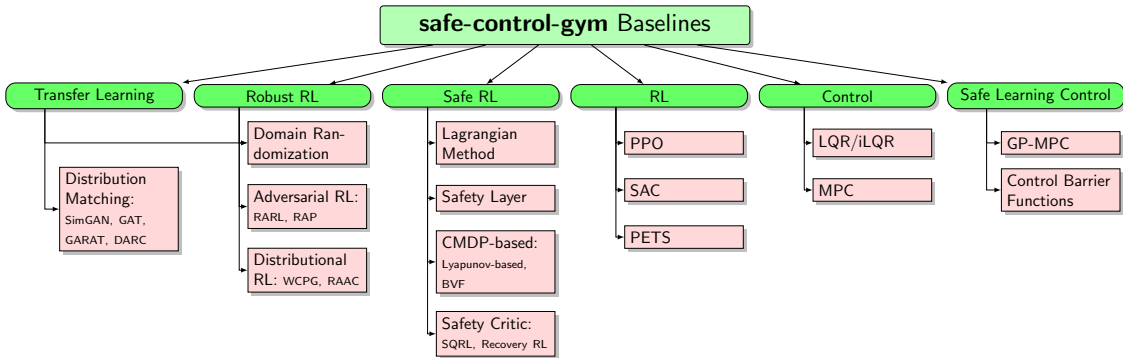


Figure 3.4: Collection of the baseline controllers in **safe-control-gym**.

## 3.4 Initial Benchmark Results

In this section, we demonstrate how **safe-control-gym** can be used to compare approaches from the control baselines discussed in Section 3.3.2, with respect to their control performance, learning efficiency, constraint satisfaction and robustness to disturbances. We use the cartpole stabilization and 2D quadrotor trajectory tracking task as our testbed. Instead of carrying out full-fledged parameter tuning for the best control or safety performance, the goal here is to highlight the potential for quantitative comparison among controllers as well as their quantitative features, showing the capability of **safe-control-gym** as a safe-learning-based benchmark suite for both RL and control approaches in robotics. Notably, **safe-control-gym** allows plotting RL and control results on a common set of axes with respect to the aforementioned performance and safety metrics, even when different controllers self-tune for different optimization objectives. To clarify, the majority of these results are also included in section 5 of our work [73].

### 3.4.1 Control Performance

In Figures 3.5 and 3.6, we show that LQR, LQR<sup>150%</sup>, GP-MPC, PPO, and SAC are able to stabilize the cartpole and track the quadrotor reference trajectory (a circle with a 1 meter radius centered at  $(0, 0, 0.5)$  on x-z plane). The LQR<sup>150%</sup> and GP-MPC controller both use a prior dynamics model with parameters overestimated by 150%. For the stabilization task, GP-MPC closely matches the closed-loop trajectory of the LQR with true parameters, albeit that its *a priori* model is the same one given to LQR<sup>150%</sup>. This shows how GP-MPC can overcome imperfect initial knowledge through learning. Both PPO and SAC yield substantially different closed-loop trajectories when compared to LQR and GP-MPC. Such difference can be due to multiple reasons: different parameterization of models (neural networks for PPO and SAC, gaussian processes for GP-MPC), different learning procedures for each algorithm framework (RL and regression), different optimization objectives tuned for each algorithm, and different prior information the algorithms have access to.

Tracking the quadrotor trajectory (Figure 3.6) results in low-frequency oscillations for PPO and SAC, a consequence possibly from learning with stochastic data and the lack of explicit long-term planning. The LQR results in relatively large tracking errors due to the inaccuracy of the linearized dynamics model in approximating the nonlinear quadrotor system, especially for aggressive maneuvers. GP-MPC’s planning horizon

and nonlinear modelling, on the other hand, effectively avoids these oscillations and tracks the reference more closely.

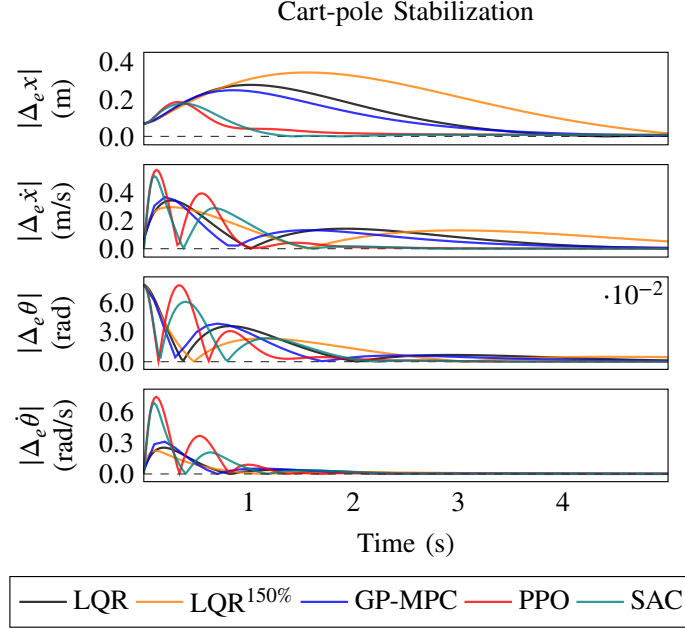


Figure 3.5: Control performance (absolute error w.r.t. reference  $\mathbf{x}^{\text{ref}}$ ,  $|\Delta_e|$ ) on the cart-pole stabilization task for different controllers and RL agents.

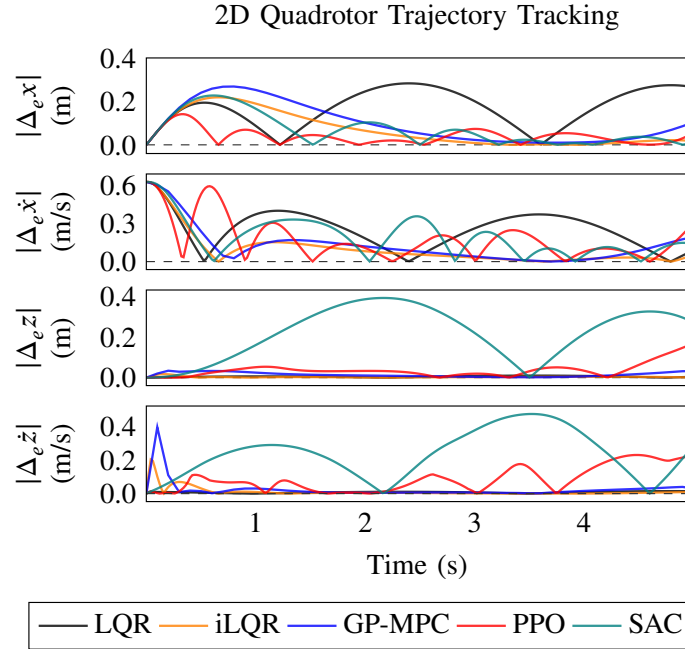


Figure 3.6: Control performance (absolute error w.r.t. reference  $\mathbf{x}^{\text{ref}}$ ,  $|\Delta_e|$ ) on the 2D quadrotor tracking task for different controllers and RL agents.

### 3.4.2 Learning Performance and Data Efficiency

Figure 3.7 shows how much data GP-MPC, PPO, and SAC require to achieve comparable trajectory root-mean-squared error (RMSE), a common metric for comparing controller performance. The values reported on the  $x$ -axis (training data) are measured in seconds of simulation time used for learning, derived as the number of elapsed environment simulation steps and PyBullet’s simulation frequency. This choice of  $x$ -axis allows controllers to use different control frequency while maintaining a common measure of the amount of learning data, for example, GP-MPC uses lower control frequency for faster inference time during planning. To enable robust, fair and reproducible comparisons, we also train or evaluate the controllers over multiple runs when applicable, and have they share the same set of randomization seeds. The plots then showcase the type of interdisciplinary comparisons enabled by **safe-control-gym**.

In both plots, the untrained GP-MPC displays a performance that is only matched after around  $10^3$  seconds of simulated data by the RL approaches. GP-MPC converges to its optimal performance with roughly one tenth (or less) of the data. This highlights how learning-based control approaches are orders of magnitude more data-efficient than model-free RL. However, this is largely the result of knowing a reasonable *a priori* model. The evaluation costs of PPO and SAC exhibit large oscillations and learning instability in the early stage, not uncommon in deep RL [36]. Once converged SAC and PPO reach a performance comparable to GP-MPC in the stabilization task. PPO can match GP-MPC on the tracking task albeit using much more data. Although optimal hyper-parameter tuning is beyond the scope at the current stage, it should be noted that, for example, PPO’s learning stability is improved by a larger batch size, especially in the quadrotor tracking task.

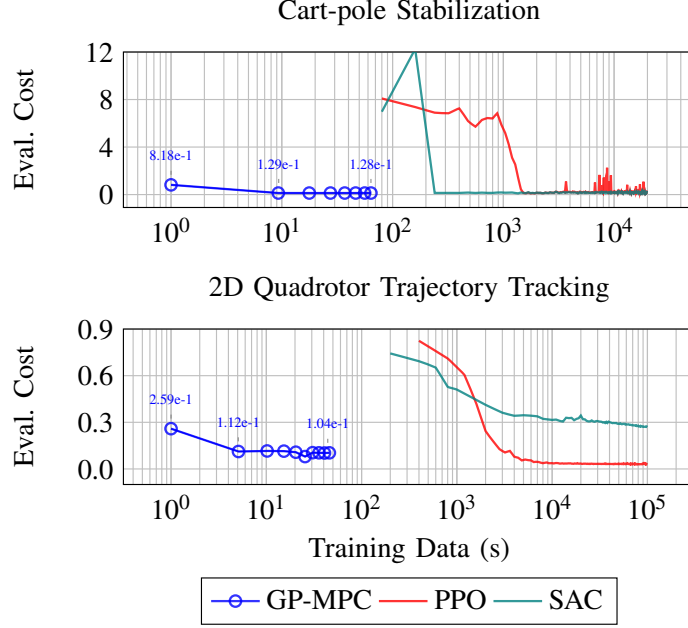


Figure 3.7: Comparison of the learning performance (on a logarithmic  $x$ -axis) of learning-based control (GP-MPC) and two RL agents (PPO, SAC). Training data is expressed as seconds of simulation time used for collection. Evaluation cost is the RMSE with respect to the target state(s). In cart-pole stabilization, the RMSE uses the full state; in 2D quadrotor tracking, the RMSE only includes the  $x$  and  $z$  component of the state.

### 3.4.3 Constraint Satisfaction

In Figure 3.8, we investigate the impact of learning and the amount of training data on the constraint violations of a learning-based controller or safe RL agent. The top plot summarizes the data efficiency of these approaches on the cartpole stabilization task. Again, leveraging the same overestimated prior model as in testing control performance, GP-MPC and the learning-based CBF require much fewer training examples to minimize the number of constraint violations than PPO with a *safety layer*. After training, GP-MPC, learning-based CBF, and safety layer PPO all achieve similar constraint satisfaction performance. Vanilla PPO also reduces the number of constraint violations but cannot match the performance of the GP-MPC and the learning-based CBF.

The bottom plot of Figure 3.8 shows reduced constraint violations for GP-MPC and PPO with a *safety layer* for the 2D quadrotor tracking task. Compared to a linear MPC with overestimated parameters, the GP-MPC meets the constraints, finding a compromise between performance and constraint satisfaction. PPO with a *safety*

*layer*, on the other hand, struggles to balance between tracking the desired trajectory and fully guaranteeing constraint satisfaction.

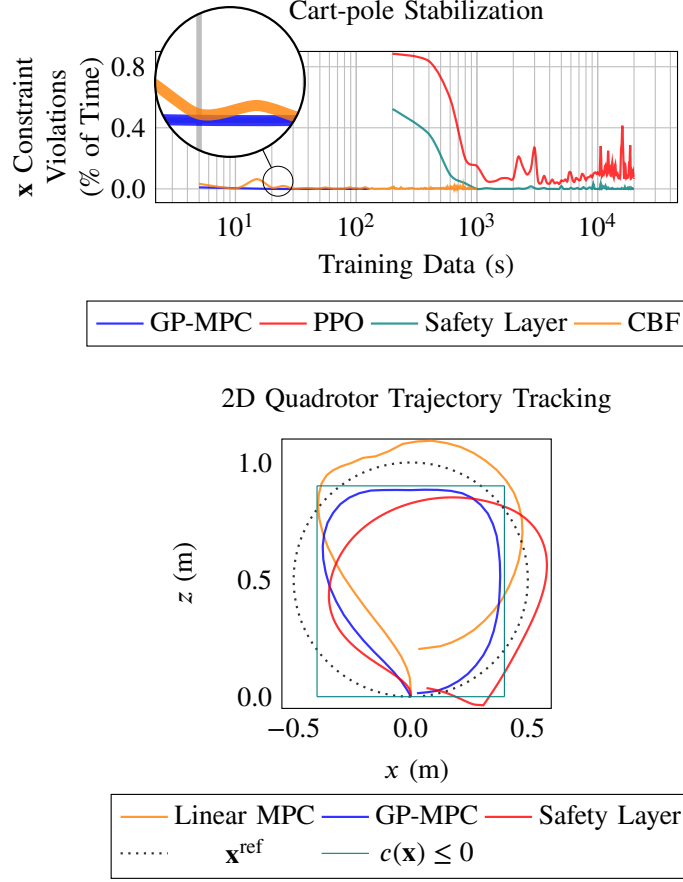


Figure 3.8: In the top plot, fraction of time spent incurring a constraint violation by learning-based control (GP-MPC), vanilla RL (PPO), safety-augmented RL (Safety Layer), and safety-certified control (CBF); in the bottom plot, trajectories for an “impossible” tracking task (with constraints narrower than the reference) for traditional control (Linear MPC), learning-based control (GP-MPC), and safety-augmented RL (PPO with Safety Layer).

### 3.4.4 Robustness

Leveraging `safe-control-gym`’s APIs for disturbance injection, Figure 3.9 shows the robustness of controllers and RL agents against parametric uncertainty (in the pole length) and white noise (on the input) for the cart-pole stabilization task.  $\text{PPO}^{\text{DR}}$  is trained with pole length randomization and improves the robustness of baseline PPO. RAP is trained against adversarial input disturbances and shows robust performance to input noise, as expected from its adversarial training. However, RAP is less robust to different classes of disturbances, such as parameter mismatch, possibly due to RAP

overfitting to the disturbances seen in training, or the lack of generalizability between input and parameter disturbances. Model-based approaches, like LQR and GP-MPC, appear to be less affected by parameter uncertainty than model-free RL. This could be due to the fact that a prior dynamics model, albeit inexact, helps generalizing across similar systems. However, LQR and GP-MPC are no less resilient to input noise, when compared to PPO or RAP. GP-MPC, in particular, might be hindered by its lower control frequency (chosen to improve computational run-time), which results in accumulation of input noise that further degrades performance.

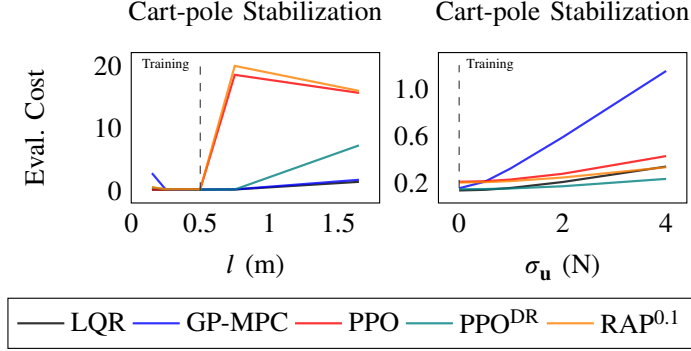


Figure 3.9: Robustness of cart-pole stabilization policies learned by traditional (LQR) and learning-based control (GP-MPC) as well as vanilla (PPO) and robust RL ( $PPO^{DR}$ , RAP) with domain randomization against perturbations in the length of the pole  $l$  (left) and a white noise disturbance  $\sim \mathcal{N}(0, \sigma_u^2)$  applied to action input  $\mathbf{u}$  (right).

### 3.5 Summary

In this chapter, we introduced **safe-control-gym**, a benchmark suite of simulation environments to evaluate safe learning-based control. In **safe-control-gym**, we combine a physics engine-based simulation with the description of the available prior knowledge and safety constraints using a symbolic framework. By doing so, we allow the development and test of a wide range of approaches, from model-free RL to learning-based MPC. We hope that **safe-control-gym** will make it easier for researchers from the RL and control communities to compare their progress, especially for the quantification of safety in robotics applications.

# Chapter 4

## Benchmark Experiments

Building on the software stack introduced in Section 3.3 and benchmark results in Section 3.4, this chapter proceeds with more in-depth experiments to benchmark the safety performance among a representative set of RL baselines. The baselines are selected according to our safe-learning survey[8] that summarizes recent development of safe control, safe RL, robust RL and more, in hope of covering a variety of safety-related scenarios during robot learning and deployment. Along with the safe-learning survey[8], the design of **safe-control-gym** [73] and the following benchmark experiments, we intend to provide a full spectrum of support for safe-learning in robotics, spanning from literature to its related tools and assets.

The benchmark experiments are divided into three categories: constraint satisfaction, robustness and transfer performance. Each features a different aspect of safety that has been proven crucial for real-life robot operations and has continuously gained momentum in recent research. Compared to Section 3.4, we expand the number of baselines to increase diversity but put a focus on model-free RL approaches due to their similarity in problem settings and training procedures, which renders the benchmark results easier to analyze. For each of the experiments, we mainly utilize the aforementioned environments from Section 3.3.1 and safe-learning features from Section 3.2. We also ablate on some of the key design choices that affect the baselines' safety performances.

Despite being tested in simulated tasks and with artificial constraints or disturbances, the benchmark results should serve as a good basis for analyzing or developing future safe control approaches. Specifically, since not all the baselines are originally developed with the same learning environments, it is expected that their performance could

differ in the **safe-control-gym** benchmark compared to their original testing environments. Nevertheless, the benchmark results still highlight the key design factors and quantitative differences among the baselines, which should be useful guidance when practitioners adapt the baselines in other new scenarios. On the other hand, the benchmark involves various evaluation tests and metrics which can be easily applied to other safe-learning approaches. This helps in the continuous development of the **safe-control-gym** benchmark, where researchers can not only quickly contribute new safe control methods, but also reuse the testing pipeline to evaluate safety performances.

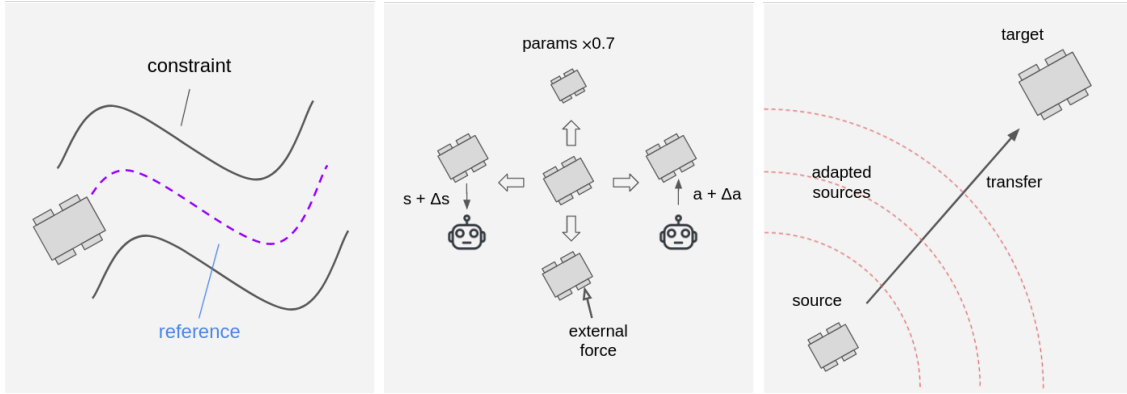


Figure 4.1: Three aspects of safety for the benchmark: *Left*: constraint satisfaction. *Mid*: robustness. *Right*: transfer performance.

## 4.1 Constraint satisfaction

Constraint satisfaction is an important measure of safe control performance and has been the subject of study for numerous safe RL works. Known challenges include how to incorporate constraints to common RL algorithms, how to mitigate constraint violations during the learning process, how to derive safety guarantees, and how to balance performance optimization and constraint satisfaction. In this section, we first compare between safety-unaware RL baselines and safety-aware RL baselines, then we compare between safe RL baselines with different formulations, lastly we look into each formulation and investigate its benefits and trade-offs.

### 4.1.1 Tasks Description

The benchmark tasks for constraint satisfaction include a angle-constrained cartpole stabilization task and a position-constrained 2D quadrotor trajectory tracking task.

Specifically for cartpole, the pole is constrained to maintain its deviation angle within a symmetric bound  $-0.2 \leq \theta \leq 0.2$ . The bound is selected to be greater than the cartpole state initialization range  $\theta_0 \in [-0.1, 0.1]$  to ensure initial feasibility; and less than the observation space bound  $|\theta| > 1.57$ . Each episode runs for 250 steps unless early termination is set, either with the system state going out of the observation space bounds or with constraint violation happening. Given the design of the constraint, it is expected that an RL agent will experience some constraint violations during training, but slowly reduce the number of violations until convergence when both optimal performance and constraint satisfaction can be achieved. For quadrotor, the drone is commanded to track a circular trajectory on the x-z plane centered at  $(0, 0, 0.5)$ , starting from near the origin. However, the drone is given a box constraint on its state  $-0.4 \leq x \leq 0.4$ ,  $-0.01 \leq z \leq 0.9$ , which partially overlaps with the circular trajectory. So it is expected to show the trade-off between improving performance and satisfying constraints for the RL baselines, differing from the cartpole task where both can be attained ideally. Each episode also runs for 250 steps and early termination can be set for training. The circular trajectory is parameterized by a continuous curve but sub-sampled to generate a way-point for each control step.

Both tasks are set to run for 5 seconds per episode with 50 Hz control frequency and 1000Hz pybullet frequency. This is inherited from Section 3.4 in consideration for control-based methods, where higher physics stepping frequency fits better with the symbolic models and lower control frequency allows reducing runtime for the expensive planning process. Environment rewards are based on the negative quadratic cost commonly used in control methods, but are exponentiated for RL to remain bounded which helps to stabilize training. Exponentiated rewards are also positive which enables the use of early termination, helping to explore the useful part of the state space more efficiently while preventing the agent from falsely learning to terminate episodes early due to accumulation of negative rewards. In addition, we use normalized action spaces that also helps in RL training; the actions output by RL agents are preprocessed (clipping and rescaling to the full actuator ranges) inside the environment before being fed to the physics engine. During evaluation, early termination is disabled and all episodes run till the full length for fair comparison among the baselines.

### 4.1.2 Baselines

The following baselines are evaluated

- **PPO [57] and SAC [33].** These two approaches are the most commonly used model-free RL baselines, and are representative in the on-policy and off-policy classes respectively. The nominal implementations do not take in account of constraints and are hence safety-unaware.
- **PPO and SAC with penalty reward.** These two variants augment the nominal implementations with reward shaping. When constraint is violated or near violation by a given tolerance value, a constant penalty is added to the reward from the environment step, which discourages RL from learning unsafe actions.
- **Safety Layer PPO [17].** The constraints are restricted to be state-based and deterministic  $c_i(s') = c_i(s, a)$  where  $s' = f(s, a)$  is the next state and  $f$  is the deterministic dynamics. It learns a linear approximation to the constraint and uses it to modify the current action  $\pi_\theta(s)$  so that constraint at the subsequent state is satisfied. Unlike the original work, our implementation uses PPO as the underlying learning agent.

$$c_i(s') = c_i(s, a) \approx c_i(s) + g_{w_i}(s)^T a$$

$$\arg \min_a \frac{1}{2} \|a - \pi_\theta(s)\|^2 \quad \text{s.t. } c_i(s, a) \leq d_i \quad \forall i$$

- **SQRL [59].** The formulation requires a safety indicator  $\mathcal{I}(s)$  where  $\mathcal{I}(s) = 1$  indicates an unsafe state, and the goal is to avoid constraint violation with probability  $\epsilon \in [0, 1]$  or in math terms  $\mathbb{E}_\pi[\mathcal{I}(s)] < \epsilon$ . A safety critic  $Q_{\text{safe}}^\pi(s, a) \approx \mathbb{E}_\pi[\mathcal{I}(s)]$  is learned during pretraining that predicts the probability of constraint violation to occur, similar to a reward critic that predicts future return. Then the safety critic projects the current policy  $\pi$  to its closest safe policy when taking actions.

$$Q_{\text{safe}}^\pi(s, a) = \mathcal{I}(s) + (1 - \mathcal{I}(s)) \mathbb{E}_{s' \sim P(\cdot|s, a), a' \sim \pi(\cdot|s')} [\gamma Q_{\text{safe}}^\pi(s', a')]$$

$$\pi_{\text{safe}} \propto \begin{cases} \pi(a|s) & \text{if } Q_{\text{safe}}^\pi \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

- **Recovery RL [62].** This approach extends SQRL by learning a separate recovery policy  $\pi_{\text{rec}}$  when constraint violation is predicted to occur. The safety critic and recovery policy are both pretrained but allows further fine-tuning

along with the current policy. The recovery policy is implemented as another SAC agent for simplicity.

$$\pi_{\text{safe}} \propto \begin{cases} \pi(a|s) & \text{if } Q_{\text{safe}}^\pi \leq \epsilon \\ \pi_{\text{rec}}(a|s) & \text{otherwise} \end{cases}$$

- **Lagrangian method.** This approach uses the Lagrange multipliers to transform RL with constraints to an unconstrained optimization problem. The Lagrange multiplier is updated jointly with policy optimization via gradient descent. The safety indicator is used for the constraint since we can reuse the pretrained safety critic from SQRL or Recovery RL; otherwise we either need to estimate  $\mathbb{E}_\pi[I(s)]$  on-policy which is high inefficient or train a separate  $Q_{\text{safe}}^\pi$  from scratch which is expected to incur more constraint violations during learning.

$$\max_{\pi} \min_{\lambda} J(\pi) + \lambda(\epsilon - \mathbb{E}_\pi[I(s)])$$

- **Lyapunov PPO [13].** Instead of a safety indicator function, the CMDP formulation is used here where the expected total constraint cost has a given threshold  $J_c(\pi) = \mathbb{E}_\pi[\sum_i \gamma^i c(s_i, a_i)] \leq d$ . This approach assumes an initial feasible baseline policy  $\pi_B$  and learns a Lyapunov function to predict the total constraint cost  $Q_{L_{\pi_B}}$ . The Lyapunov value function satisfies feasibility at the initial state and contraction under the Bellman operator, hence to ensure constraint satisfaction at all time during policy update. Given continuous state-action space, such property is then shown as a state-based constraint  $Q_{L_{\pi_B}}(s, \pi_\theta(s)) - Q_{L_{\pi_B}}(s, \pi_B(s)) \leq \epsilon = (1 - \gamma)(d - J_c^{\pi_\theta}(s_0))$ ,  $\forall s$  where  $\epsilon$  is the maximum constraint budget left for policy improvement. For implementation, we adapt PPO as the learning agent and use policy from the previous update iteration as the feasible policy as suggested in the original work for policy gradient methods.

$$\arg \min_a \frac{1}{2} \|a - \pi_\theta(s)\|^2 \quad \text{s.t.} \quad Q_{L_{\pi_B}}(s, \pi_\theta(s)) - Q_{L_{\pi_B}}(s, \pi_B(s)) \leq \epsilon$$

- **BVF [56].** Similar to Lyapunov PPO, BVF is also based on CMDP and the idea of breaking down a trajectory-based constraint into single-step state dependent constraints. Besides the usual (forward) constraint value function  $Q_c^\pi$

that predicts cost-to-go, it additionally learns a backward constraint value function  $\overleftarrow{V}_c^\pi$  that captures cost occurred so far. So the total constraint cost can be approximated as  $J_c(\pi) = Q_c^\pi(s, a) + \overleftarrow{V}_c^\pi(s) - c(s)$  at any state. Note the current constraint cost here is incurred from the previous state-action step. For implementation we build on SAC where the value functions can be learned jointly. A safe action can be constructed similar to the Safe Explorer work but with the decomposed constraint.

$$\arg \min_a \frac{1}{2} \|a - \pi_\theta(s)\|^2 \quad \text{s.t.} \quad Q_c^\pi(s, a) + \overleftarrow{V}_c^\pi(s) - c(s) \leq \epsilon$$

### 4.1.3 Results

The aforementioned baselines are trained and evaluated on the benchmark tasks with respect to three metrics of performance and safety: RMSE, failure rate and constraint violations. Similar to Section 3.4, we compute the average root mean square error of test episodes as the common performance metric. For safety, failure rate represents the ratio of test episodes with any constraint violation, while the constraint violations metric represents the average number of constraint violation steps normalized by episode length; constraint violations can provide a more detailed measure of constraint satisfaction than failure rate.

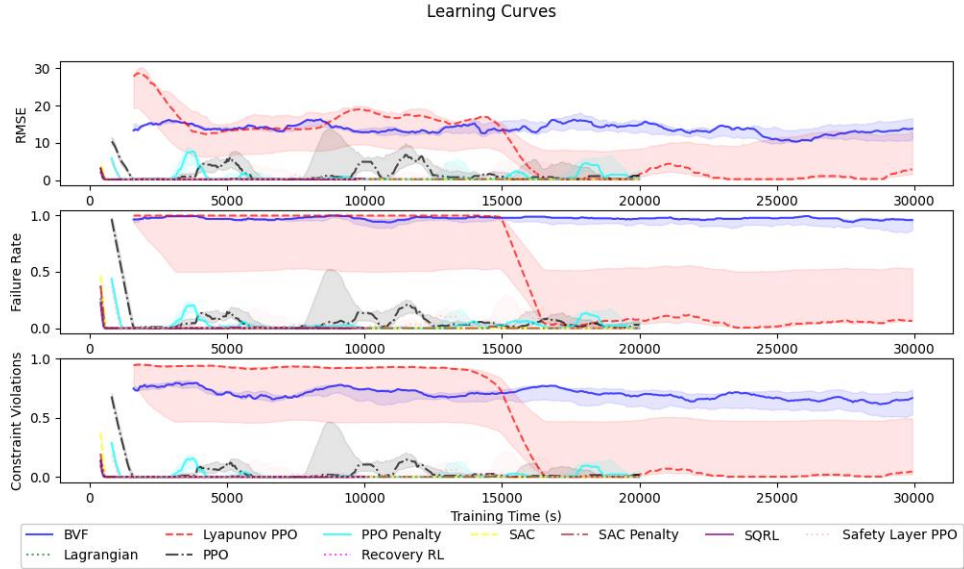


Figure 4.2: Learning curves of safe control baselines on the cartpole stabilization task with bounded angle constraint.

To begin with, we show the learning performance of the baselines in Figure 4.2 and

4.3. In the relatively simple cartpole task, most controllers learn to stabilize the pole while staying within the constrained angle bounds. Two noticeable distinctions are the CMDP-based methods Lyapunov PPO and BVF. Lyapunov PPO has a slower learning process but eventually converges to low RMSE and constraint violations on par with other baselines. On the other hand, BVF struggles to learn the task and barely shows improvement over the given metrics. This can be the result of suppressing the CMDP threshold  $J_c(\pi) \leq d$ , which is set to one  $d = 1$  to encourage almost zero constraint violation. Such formulation ideally makes the CMDP baselines comparable to other safe baselines that only optimizes for no constraint violations, but could lead to overly conservative behaviors and even hinder learning. By contrast, PPO and Safety Layer PPO have decent performances despite occasional constraint violations during learning. This is expected for PPO since both the state initialization range and goal state are within the feasible set, so an optimal PPO controller should not incur many violations although being unaware of constraints. Other baselines including simple safety augmented baselines (PPO/SAC Penalty, Lagrangian method) and safety-critic baselines (SQRL, Recovery RL) all show fast learning and close to zero constraint violation towards convergence.

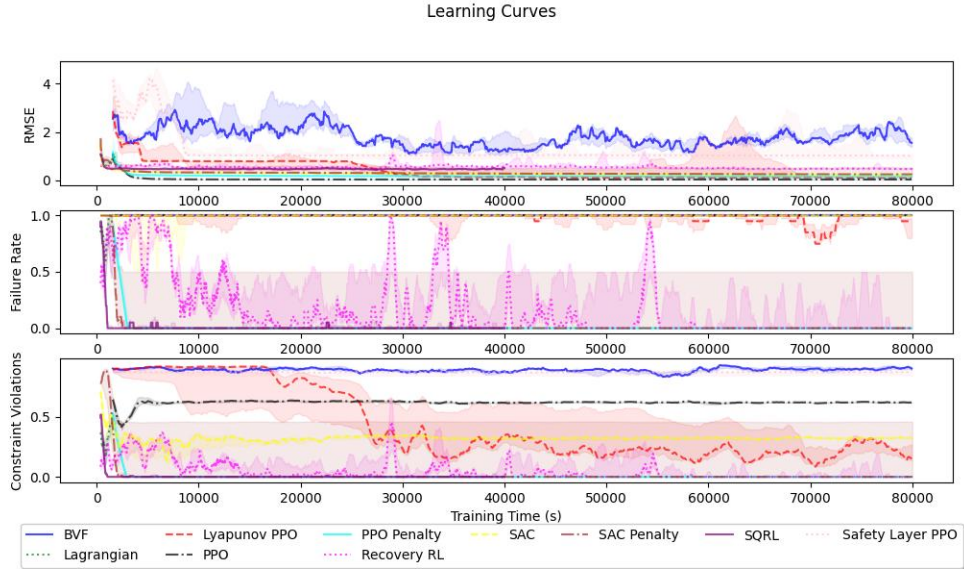


Figure 4.3: Learning curves of safe control baselines on the quadrotor tracking task with bounded position constraint.

In the harder quadrotor tracking task, the reference trajectory states are not entirely within the constraint feasible set, so the safe baselines need to trade-off between performance and constraint satisfaction. Based on Figure 4.3, it is observed that

constraint-unaware baselines (PPO, SAC) have the best RMSE but they always violate constraints which is undesired in safety-critical context. Baselines with simple reward penalty prove to be effective, reaching low RMSE and low constraint violations quickly during training, showing improvement on safe learning compared to the constraint-unaware baselines. However, SAC Penalty shows a large deviation in the learning curves indicating potential instability on more complex tasks. For baselines that utilize a safety critic (pretrained and then shared in learning), SQRL and Recovery RL have relatively low RMSE and constraint violations. Specifically, Recovery RL show drastic oscillations in its learning curves, possibly due to the continual fine-tuning of the safety critic and recovery policy that affects learning stability. Safety Layer PPO performs poorly compared to other baselines, since it relies on pre-training constraint models with uniformly random actions while the tracking task has sequential goals, this could lead to inconsistent exploration and inaccurate constraint models, hindering the subsequent policy training. For the CMDP-based methods, Lyapunov PPO achieves comparable RMSE to other baselines and better constraint violations than constraint-unaware baselines but not the others. BVF still underperforms in either RMSE or the safety metrics, showing difficulty in learning the task as well as satisfying constraints given a tight constraint threshold. The BVF implementation is based on SAC and learns two additional value functions concurrently, but SAC has been shown difficult in learning the tracking task compared to PPO in Section 3.4; hence it might also explain BVF’s poor performance here.

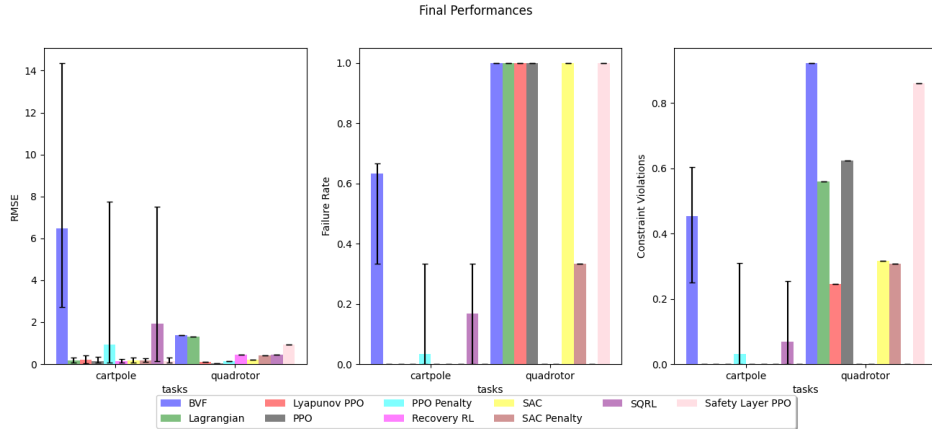


Figure 4.4: Final test performances among all baselines and tasks.

In Figure 4.4, we summarize the final test performances with the three metrics of all the baseline controllers and tasks. Most baselines except BVF learn to complete the cartpole stabilization task while keeping the pole angle within constraint bounds.

However, in the quadrotor tracking task with conflict between task performance and constraint satisfaction, only PPO Penalty, SQRL and Recovery RL manage to balance the two objectives. Other baselines show varying levels of constraint violation.

#### 4.1.4 Ablations

Given the benchmark results presented above, it is observed that the baselines have different performance trends and they vary depending on the task or task constraints. To investigate what are the important factors that affect their performances, in this section we conduct ablation studies on the several classes of safe RL approaches respectively. The ablations should provide useful tools to analyze the methods as well as guidance on any further tuning.

##### Safety critic

For safety critic based methods (SQRL, Recovery RL), they often require a safety threshold  $\epsilon \in [0, 1]$  that determines when to take actions to optimize task performance and when to take safe actions to avoid forthcoming constraint violation. Therefore, we benchmark the safety critic methods with varying safety thresholds. Since both their implementations are based on SAC, we also add SAC and the simple SAC Penalty baseline for comparison.

Table 4.1 shows the comparison results in the cartpole task. Since all the methods perform well with zero constraint violation given the default constraint  $-0.2 \leq \theta \leq 0.2$ , two harder constraints are added for finer comparison. The mid-level constraint  $-0.07 \leq \theta \leq 0.07$  has a tighter bound than the default constraint, requiring more aggressive control to keep the pole safe. The hard constraint  $-0.05 \leq \theta \leq 0.05$  imposes angle bounds that are the same as the state initialization bound  $\theta_0 \in [-0.05, 0.05]$ , which means the controller must be responsive and kept safe from the start of episode. The nonzero constraint violation of SAC demonstrates the necessity of learning with constraints as expected. The safety critic methods prove to be more effective given harder constraints, showing equal or better safety performances than simple reward penalty method. Also it is observed that threshold  $\epsilon = 0.2$  has an overall better safety performance in both SQRL and Recovery RL. Since the threshold controls the allowable probability of failure, this result confirms the need to add proper buffer for constraint violation in safety critic methods, as can be seen that either low threshold  $\epsilon = 0.1$  or high threshold  $\epsilon = 0.4$  leads to noticeable increase in failure rate and

constraint violations.

| Metrics                                          | SAC   | SAC<br>Penalty | SQRL  |       |       |       | Recovery RL |       |       |       |
|--------------------------------------------------|-------|----------------|-------|-------|-------|-------|-------------|-------|-------|-------|
|                                                  |       |                | 0.1   | 0.2   | 0.3   | 0.4   | 0.1         | 0.2   | 0.3   | 0.4   |
| constraint (default) $-0.2 \leq \theta \leq 0.2$ |       |                |       |       |       |       |             |       |       |       |
| RMSE                                             | 0.178 | 0.18           | 0.168 | 0.167 | 0.158 | 0.16  | 0.164       | 0.158 | 0.158 | 0.184 |
| Failure rate                                     | 0.    | 0.             | 0.    | 0.    | 0.    | 0.    | 0.          | 0.    | 0.    | 0.    |
| Constraint violations                            | 0.    | 0.             | 0.    | 0.    | 0.    | 0.    | 0.          | 0.    | 0.    | 0.    |
| constraint (mid) $-0.07 \leq \theta \leq 0.07$   |       |                |       |       |       |       |             |       |       |       |
| RMSE                                             | 0.186 | 0.151          | 0.822 | 0.139 | 2.916 | 0.856 | 0.148       | 0.145 | 0.146 | 0.153 |
| Failure rate                                     | 0.617 | 0.217          | 0.25  | 0.217 | 0.283 | 0.233 | 0.283       | 0.25  | 0.35  | 0.283 |
| Constraint violations                            | 0.03  | 0.002          | 0.02  | 0.003 | 0.065 | 0.02  | 0.003       | 0.003 | 0.003 | 0.002 |
| constraint (hard) $-0.05 \leq \theta \leq 0.05$  |       |                |       |       |       |       |             |       |       |       |
| RMSE                                             | 0.183 | 0.15           | 3.371 | 4.877 | 0.468 | 5.524 | 0.152       | 0.142 | 0.261 | 0.155 |
| Failure rate                                     | 0.733 | 0.467          | 0.467 | 0.367 | 0.4   | 0.45  | 0.55        | 0.5   | 0.467 | 0.65  |
| Constraint violations                            | 0.056 | 0.006          | 0.084 | 0.186 | 0.045 | 0.176 | 0.008       | 0.005 | 0.006 | 0.01  |

Table 4.1: Ablation on safety critic-based methods with varying safety thresholds and difficulty of constraints in the cartpole task. Highlighted cell shows the best result in the row.

In addition, we evaluate the quality of the learned safety critics by plotting the distribution of their predicted probability of failure on test trajectories in Figure 4.5. The trajectories are generated using a  $\epsilon$ -greedy strategy, namely by mixing actions from the learned policy and a uniform random policy. It can be observed that in subfigure (A) (B) for the cartpole task, the predicted values stay low when the trajectories are in the feasible set and quickly increase as the trajectories approach the constraint bounds. Also in region where a trajectory goes out of the bound but eventually returns to the feasible set, the predicted values also transition from high to low. This indicates successful learning of the safety critic, which is expected to capture the probability of failure in future rollouts conditioned on the current state. In subfigure (C) (D) for the quadrotor task, we increase the portion of random actions to include more unsafe test trajectories. Similar to (A) (B), safety critic outputs lower values when trajectories are in the feasible set than in the unsafe region. But it is also observed that in region where the reference lies outside of the constraint bound and induces a conflicting scenario, the safety critic tends to make optimistic predictions by assigning low values to unsafe states.

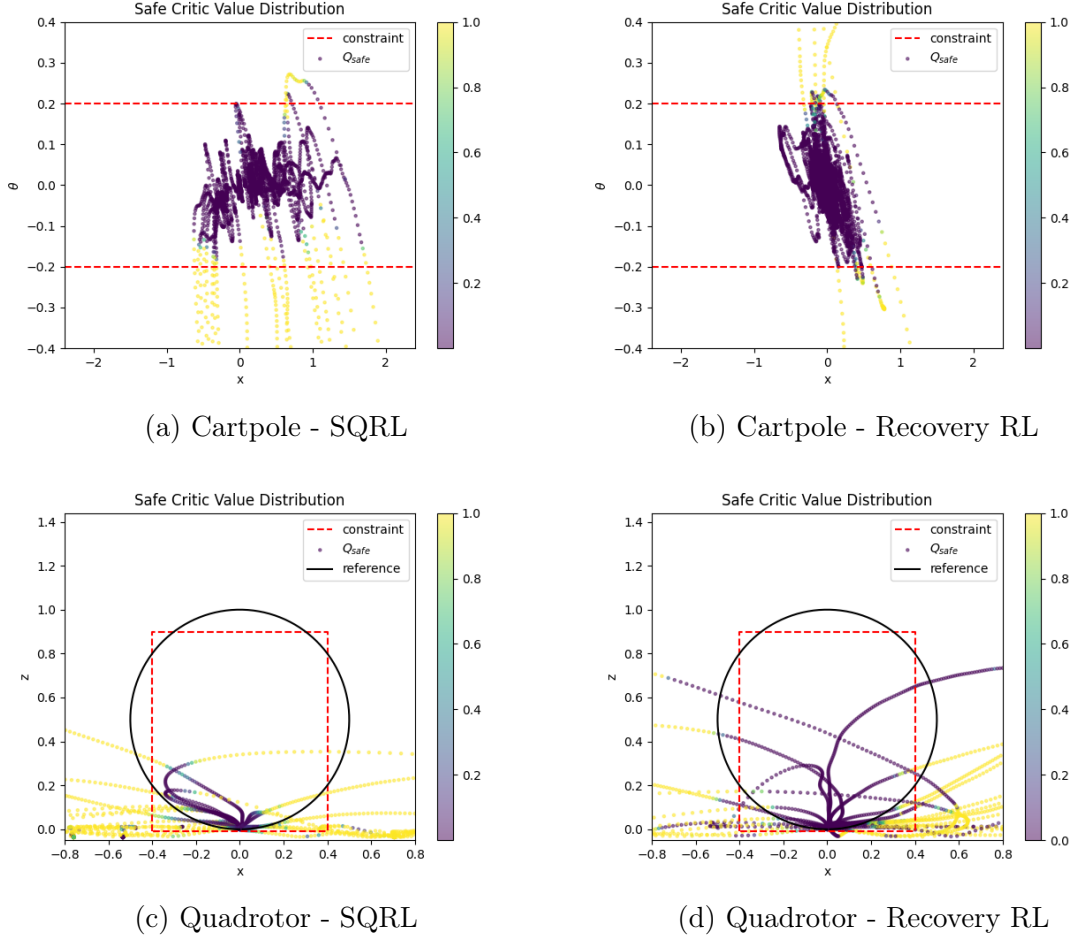


Figure 4.5: Distribution of safety critic predicted values on trajectories generated from a  $\epsilon$ -greedy policy.

### CMDP thresholds

In the benchmark results, CMDP-based methods show relatively poor performances compared to other baselines, and it is suspected that the CMDP constraint threshold plays an important role. To investigate this, we ablate against different thresholds for Lyapunov PPO and BVF. Since both of them are implemented based on PPO, the PPO and PPO Penalty baselines are also added for comparison. The ablation results are shown in Table 4.2. When the constraint threshold is relaxed  $d > 1$ , both Lyapunov PPO and BVF show an improvement in safety performance, although in theory the relaxed CMDP will allow more constraint violations. This effect is particularly evident for Lyapunov PPO with threshold  $d = 10$  and BVF with threshold  $d = 20$ , whose resulting failure rate and constraint violations are on par with or even better than the reward penalty baseline. It could indicate the necessity of having

proper buffer for constraint violations similar to the safety critic threshold ablation, especially for BVF where aggressive threshold  $d = 1$  and loose threshold  $d = 30$  both lead to drastically worst performance. To further look into this issue, we implement a variant of BVF (BVF - schedule) that uses varying threshold in training, scheduled to linearly decrease from 20 to 1 in the first half of the learning steps and kept as 1 in the latter half. The variant should effectively alleviate the conservatism from imposing aggressive threshold in early training. Although the variant shows slightly better performance than using a constant threshold 1, it is no better than using the other constant thresholds.

| Metrics               | PPO   | PPO<br>Penalty | Lyapunov PPO |       |       |       | BVF    |       |       |        |          |
|-----------------------|-------|----------------|--------------|-------|-------|-------|--------|-------|-------|--------|----------|
|                       |       |                | 1            | 10    | 20    | 30    | 1      | 10    | 20    | 30     | schedule |
| RMSE                  | 0.23  | 1.292          | 21.339       | 0.224 | 0.646 | 0.308 | 10.221 | 3.191 | 0.518 | 11.012 | 8.702    |
| Failure rate          | 0.057 | 0.06           | 0.687        | 0.027 | 0.103 | 0.023 | 0.937  | 0.383 | 0.057 | 0.363  | 0.66     |
| Constraint violations | 0.002 | 0.051          | 0.641        | 0.001 | 0.021 | 0.006 | 0.61   | 0.284 | 0.016 | 0.326  | 0.515    |

Table 4.2: Ablation on CMDP based methods with varying constraint thresholds in the cartpole task. Highlighted cell shows the best result in the row.

### Safety layer models

Besides the safety critic and CMDP methods, another distinct baseline is the Safety Layer PPO, which learns explicit models to approximate the constraint functions and then uses the models in an action filter. Naturally, the accuracy of the learned constraint models are crucial for safe performance. In Figure 4.6, we plot the distribution of the constraint model prediction errors on test trajectories. For subfigure (A), larger errors are observed in region outside and near the constraint while smaller errors are concentrated around the stabilization goal state within the feasible set. This can be the consequence of two factors: the goal state is within the state initialization range as well as the feasible set, and the constraint models are pretrained with uniformly random actions. Hence, there is sufficient exploration and the induced training data distribution captures the constraint-critical regions well. Subfigure (B) shows a similar trend but seems more noisy in the error distribution. But different to (A), there is a lack of consistent exploration due to the random exploration strategy and the tracking task involves sequential goals. This can be seen from the heavy cluster of errors around the starting point and few data points around upper part of the state space. Consequently, it leads to poor accuracy of the learned constraint models in region near farther goals of the reference, and hurts learning of the downstream controller

using the action filter.

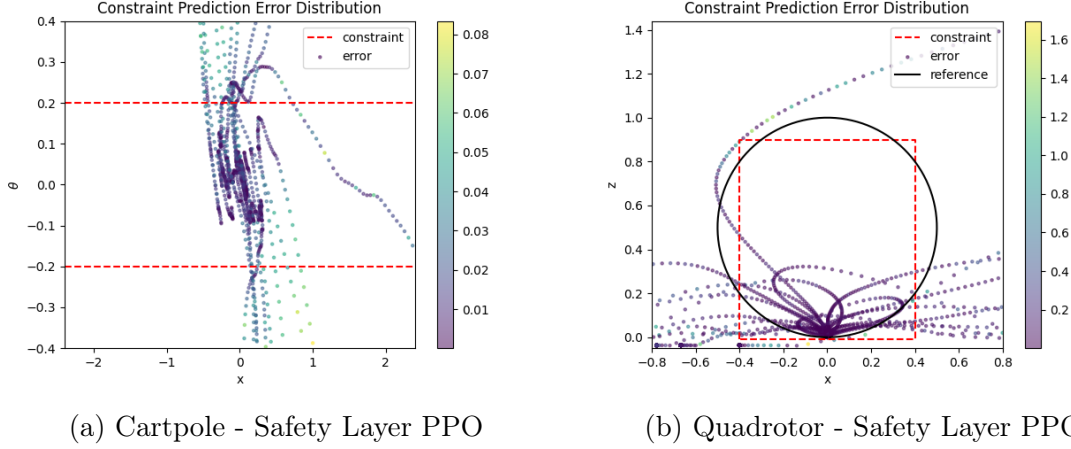


Figure 4.6: Distribution of the constraint model prediction mean-square-error on trajectories generated from a  $\epsilon$ -greedy policy.

## 4.2 Robustness

Robust control and robust RL both try to address control in the presence of disturbances. In real life, disturbances are prevalent and come in different types such as measurement noise and unmodelled dynamics; they are critical factors to consider for safe robot operation, not to mention for data-driven methods like RL where imperfect knowledge is involved. Besides, robustness is closely connected to the notion of generalization that the RL community has long sought for as one of the most desired properties. It is hence crucial to include robustness as a performance or safety metric for our benchmark. In this section, we benchmark several popular frameworks of robust RL; we compare their learning performances and safety profiles with respect to a variety of disturbance tests. Similar to Section 4.1, we also conduct ablations to investigate some key design factors of these frameworks.

### 4.2.1 Tasks Description

The benchmark tasks for robustness are also based on the cartpole stabilization task and 2D quadrotor trajectory tracking task. To benchmark for robustness, the disturbance API of **safe-control-gym** is utilized to create artificial nonidealities in the environments for both training and evaluation. Concretely, we first have the common observation and action noises modelled as uniform or normal distributions with vary-

ing scales. The dynamics disturbance are environment-specific; for cartpole it is a 2D random tapping force acting on the center-of-mass of the pole and for quadrotor it is another random force acting on the center-of-mass of the drone. Besides disturbance noise, another important class of nonideality is system parameter mismatch. The disturbance API allows setting the system parameters upon the start of episodes, either as fixed values or distributions; for the latter case a random set of parameters will be sampled per episode which supports implementing domain randomization. The changeable parameters are also environment-specific; for cartpole they consist of the pole length, pole mass and cart mass and for quadrotor they include the drone mass and moment of inertia. Moreover, these nonidealities can be combined and configured via setting the configuration files to create harder scenarios. Other settings such as episode length, reward and action space are kept the same as tasks in Section 4.1.

### 4.2.2 Baselines

The following baselines are evaluated

- **PPO [57] and SAC [33]**. Similar to Section 4.1, these two are used as general baselines for comparison. But different to being unsafe due to constraint-agnostic, PPO and SAC could have certain level of intrinsic robustness, for example, the maximum entropy formulation in SAC.
- **Domain Randomization (DR) [55, 64, 10]**. This approach can be built on top of any standard RL algorithm. The key idea is to generate a set of training environments with randomized system parameters, forcing the agent to generalize over these environment variants and hopefully even to environments out of the training distribution. Typically, DR requires careful selection of system parameters and tuning of the randomization ranges.
- **RARL [51]**. Combining robust control and adversarial training, this approach optimizes for worst-case performance under a given family of disturbances. The disturbance is realized as a separate learning agent or adversary, trained jointly with the main agent or protagonist but using the opposite reward. This forms a max-min optimization and a learning curriculum where the protagonist improves control performance against harder disturbances. Concretely in RARL, disturbance is implemented as an external force to the system. Both the protagonist

and adversary are parameterized as a PPO agent.

$$\max_{\pi} \min_{\pi_{\text{adv}} \in \mathcal{F}} J(\pi, \pi_{\text{adv}})$$

- **RAP [68]**. This approach extends RARL by introducing multiple adversaries, with the reasoning that the protagonist could overfit to a single adversary or be falsely misled to train on unrealistic disturbances. Different to RARL, RAP implements disturbance as action noise and before each training episode starts, a random adversary is selected to produce disturbance action at each step. All adversaries are implemented as PPO.

$$\max_{\pi} \min_{\{\pi_{\text{adv}}\}_n \in \mathcal{F}} J(\pi, \{\pi_{\text{adv}}\}_n)$$

- **WCPG [60]**. A different formulation in robust RL that optimizes for worst-case performance is to model the full distribution of returns as in distributional RL, and use a risk-averse measure on the distribution as the optimization objective. In WCPG, the return distribution is parameterized as Gaussians  $Z^{\pi}(s, a) \sim \mathcal{N}(Q^{\pi}(s, a), \Upsilon^{\pi}(s, a))$ , where the mean and variance are both value functions to be learned with TD methods. With this parameterization, conditional value at risk (CVaR) of the returns can be computed in closed-form, which captures the average of low-percentile returns as a surrogate to worst-case performance. Given a percentile threshold  $\alpha$ , the actor then uses CVaR of the returns  $\Gamma^{\pi}(s, a, \alpha)$  in its loss gradient. We implement WCPG based on SAC with the new update rules for the return distribution variance and actor.

$$\begin{aligned} \Gamma^{\pi}(s, a, \alpha) &= \text{CVaR}_{\alpha} = Q^{\pi}(s, a) - (\phi(\alpha)/\Phi(\alpha))\sqrt{\Upsilon^{\pi}(s, a)} \\ \phi(\cdot), \Phi(\cdot) &- \text{PDF and CDF of standard normal distribution} \\ \nabla_{\theta} J_{\alpha} &= \mathbb{E}_{\pi}[\nabla_{\theta} \log \pi_{\theta}(a|s) \Gamma^{\pi}(s, a, \alpha)] \end{aligned}$$

- **RAAC [66]**. Similar to WCPG, this approach optimizes for the CVaR of returns but parameterizes the return distribution as value functions conditioned on the percentile  $Z^{\pi}(s, a; \tau)$ . These value functions can also be learned via TD methods and be used to approximate the returns' CVaR. The original work focuses on offline RL setting but we adapt the online learning version called RAAC for comparison. Again our implementation is based on SAC with the

additional update rules for the value function and actor.

$$\begin{aligned} \text{CVaR}_\alpha(Z^{\pi_\theta}(s, a; \tau)) &= \frac{1}{\alpha} \int_0^\alpha Z^{\pi_\theta}(s, a; \tau) d\tau \\ &\approx \frac{1}{\alpha K} \sum_{k=1}^K Z^{\pi_\theta}(s, a; \tau_k), \quad \tau_k \sim \mathcal{U}(0, \alpha) \end{aligned}$$

### 4.2.3 Results

The robust baselines are evaluated on the benchmark tasks using RMSE as the main metric. The learning curves are shown in Figure 4.7 and 4.8 below. However, different baselines utilize different disturbances from **safe-control-gym** to aid their training, so the learning performance might not be informative for fair comparison. Therefore, we also benchmark their final performances against multiple disturbance scenarios to showcase the different aspects of robustness.

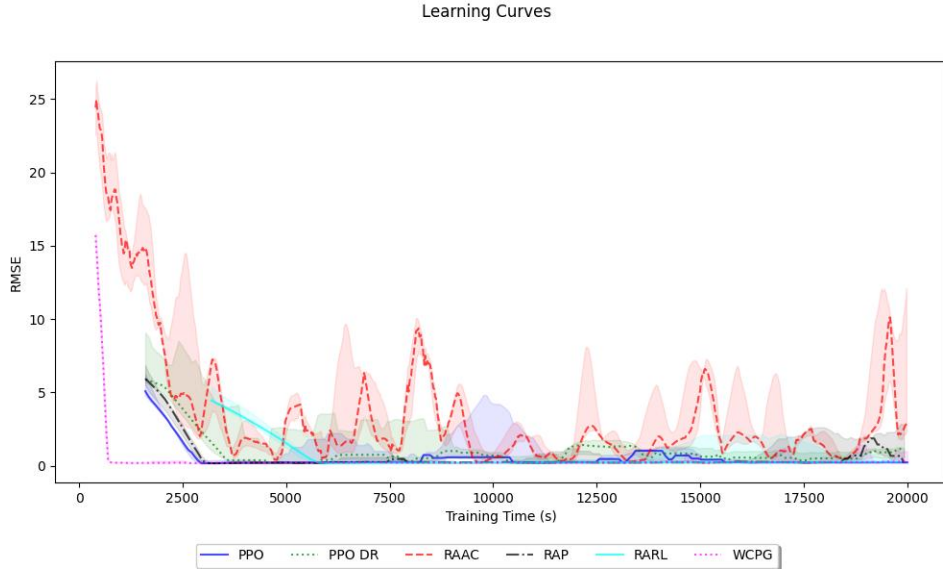


Figure 4.7: Learning curves of the robust baselines on the cartpole stabilization task.

In the cartpole task from Figure 4.7, it is observed that most robust baselines show increase in performance and converge to close-to-zero RMSE with the exception of RAAC, which exhibits large oscillations in its learning curve and does not stabilize in the given training time frame of the other baselines. WCPG has a noticeable faster convergence rate, followed by PPO, RAP and RARL in order. PPO and RAP have similar learning trends while RARL requires more data to converge. This is reasonable since RAP is trained with action noise, but RARL is trained with an external tap force

on the pole’s center-of-mass which intuitively poses a harder challenge to stabilize. PPO with domain randomization (PPO DR) experiences an initial slow learning phase with high variance in RMSE then the more stable performance trend till the end of training, which can be attributed to the need of sufficient exploration given the randomized training environments.

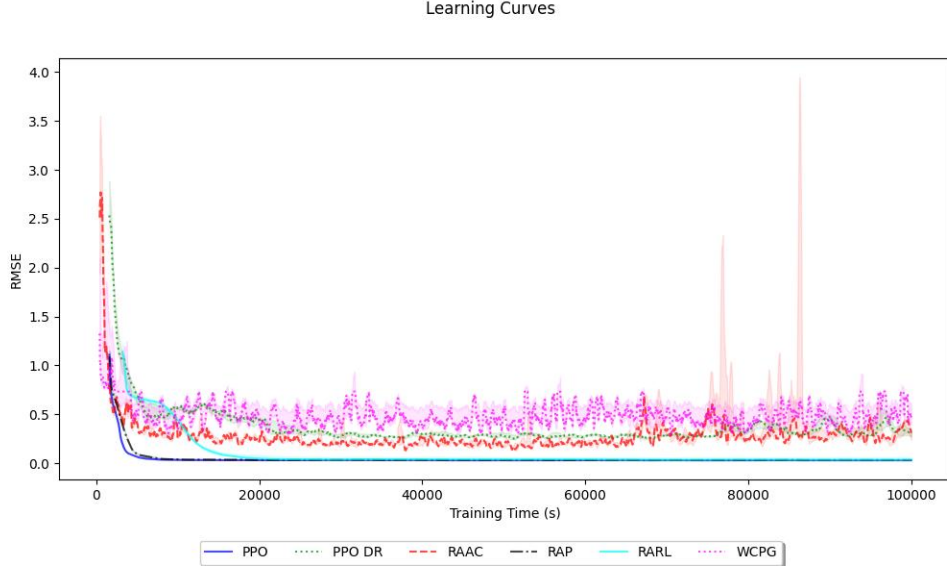


Figure 4.8: Learning curves of the robust baselines on the quadrotor tracking task.

In the quadrotor task from Figure 4.8, again PPO and the two adversarial RL methods (RAP, RARL) converge to low RMSE and show similar learning efficiency trends as in the cartpole task. The two distribution RL based methods (WCPG, RAAC) however underperform in RMSE compared to PPO and adversarial methods. Additionally, they show high frequency oscillations in the learning curves. This effect might be due to several factors: (i) reliance on sampling-based gradients for learning which is noisy; (ii) learning value functions that capture the full distribution instead of the mean of returns which has higher complexity and hence requires more samples. Combining these factors, it is expected that the learning performance will have high variance as reflected by the oscillations. WCPG does not exhibit such effect in cartpole stabilization, possibly because of the lower task difficulty and hence quick convergence to the optimal policy. Besides, WCPG has larger oscillation magnitudes than RAAC which is due to the difference in their training setup. Specifically, RAAC uses a fixed alpha threshold to optimize the  $\text{CVaR}_\alpha$  of returns but WCPG uniformly samples a new alpha value from a predefined range for each training episode; the latter obviously induces higher stochasticity for learning. Lastly, PPO DR also shows a

slow learning trend at the initial stage similar to cartpole, but it has a stagnant learning curve afterwards and fails to converge to low RMSE like PPO, indicating possible local minima. This could emphasize the difficulty in tuning domain randomization parameters in more complex tasks.

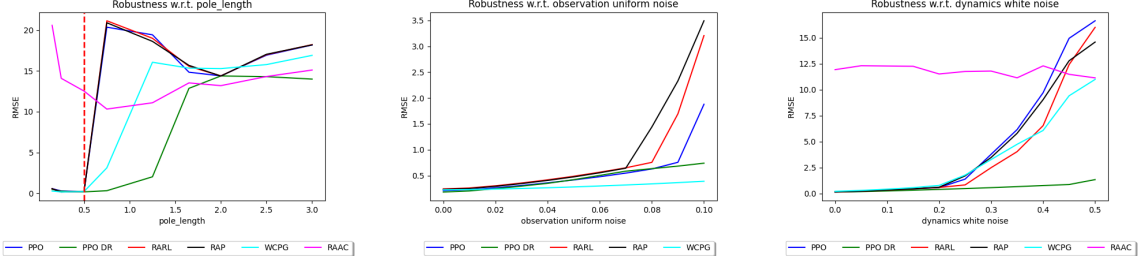


Figure 4.9: Robustness performance benchmark with one varying disturbance in cartpole: *Left*: pole length (the vertical dash line represents default pole length in training environment). *Mid*: observation noise (RAAC is not plotted due to bad performance curve lying out of range from the others). *Right*: external tap force on pole center-of-mass.

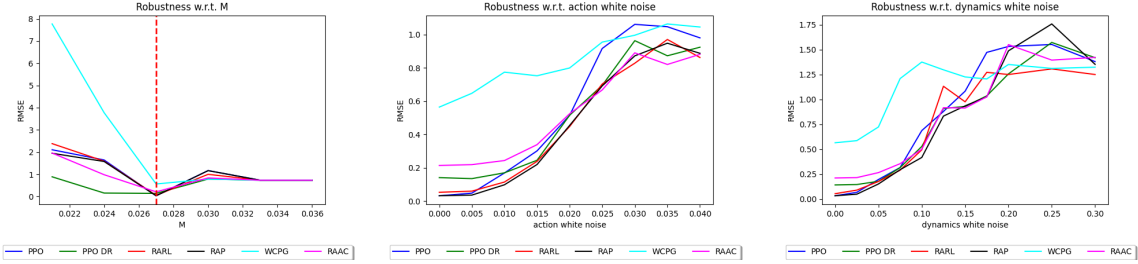


Figure 4.10: Robustness performance benchmark with one varying disturbance in quadrotor: *Left*: drone mass (the vertical dash line represents default drone mass in training environment). *Mid*: action noise. *Right*: external 2D force on drone.

The above benchmark results shed light on features of learning process among the robust baselines. To look at their final test performances against disturbances, we also benchmark them with different classes of disturbances and for each class, we evaluate the baselines given different severity of the disturbance.

Figure 4.9 shows the baseline performances in cartpole with respect to three kinds of disturbances: (i) varying pole lengths; (ii) observation noise from uniform distribution of varying scales; and (iii) external 2D tap force on pole’s center-of-mass from normal distribution of varying scales. It is observed that PPO DR has the best overall performance across most of these disturbances, proving its efficacy in learning robust policy

despite simplicity of the method. The two adversarial RL baselines (RARL, RAP) have similar performance trends to PPO with these disturbances, except RARL shows slight advantage with external force after scale 0.3, which is expected since RARL is formulated to train against adversarial external forces. Similarly, another related benchmark test from Section 3.4 Figure 3.9 also validate RAP to have a more robust performance with action noise, given RAP is formulated to train against it. These two results may indicate the limited generalizability of adversarial RL methods to other types of disturbances when used alone, alternatively it may suggest to train adversarially against a hybrid set of disturbances to attain broader robustness. Unlike the adversarial methods that require predefined disturbances in the training environment, the distributional RL methods (WCPG, RAAC) optimize CVaR of the returns without prior knowledge on any disturbance. WCPG shows comparable robustness to the adversarial methods and better performance in certain regions such as small variation of pole length or high perturbation of external force. Noticeably, WCPG is even outperforming PPO DR when facing observation noise. RAAC is seemingly nonrobust against the three tested disturbances on average, except in the left plot where RAAC shows better RMSE than some other baselines when pole length variations are large. RAAC’s underperforming compared to WCPG can be the result of using more complex parameterization for the distributed value function as well as using a fixed CVaR threshold throughout training.

Moving on to the harder quadrotor tracking task in Figure 4.10, PPO DR no longer shows obvious advantage to other baselines across the tested disturbances, except in the low quadrotor mass region in the left plot. By contrast, PPO, RAP and RARL again have similar performance trends, specifically RAP and RARL outperform PPO in their own respective disturbance domains where they are formulated to train against. Another difference is the robustness performance of WCPG and RAAC. Given the harder task, RAAC starts to outperform WCPG in all these disturbances and is competitive to the adversarial RL baselines. WCPG falls short in this harder task, placing behind other baselines and only becomes comparable with large enough disturbances. This is possibly due to the benefit of more representation power, since WCPG parameterizes the return distribution only as Gaussian while RAAC parameterizes it as quantiles that ideally can capture broader distributions, making RAAC more robust to complex changes to the environment.

To further investigate how the trained robust baselines perform in the presence of hybrid disturbances, in the following we show the benchmark results by evaluating

the baselines with respect to two varying disturbances.

Figure 4.11 shows a set of performance heatmaps for each baseline with varying pole lengths and pole masses in the cartpole task. By looking at the scale and distribution of the heatmap values, it can be concluded that PPO DR has the overall most robust performance against system parameters change. This result is consistent with how domain randomization is formulated similar to RAP’s and RARL’s better performances against action noise and external force respectively. Besides, WCPG show better robustness to system parameter change compared to other baselines. PPO, RARL and RAP have similar heatmap patterns, which indicates similar performances in terms of robustness to system parameters. Due to suboptimal learning as shown in Figure 4.7, RAAC’s heatmap shows higher RMSE on average and a noisier distribution compared to other baselines. It is also observed that most baselines except RAAC have some analogous patterns: (i) scaling both system parameters results in lower RMSE than scaling one parameter along as seen from the bottom left and upper right corners in the heatmaps; (ii) pole length has higher sensitivity than pole mass as seen from the large RMSE change column-wise than row-wise, and (iii) lower pole lengths seem easier for control than higher ones with most of the low RMSE values concentrated in the bottom region.

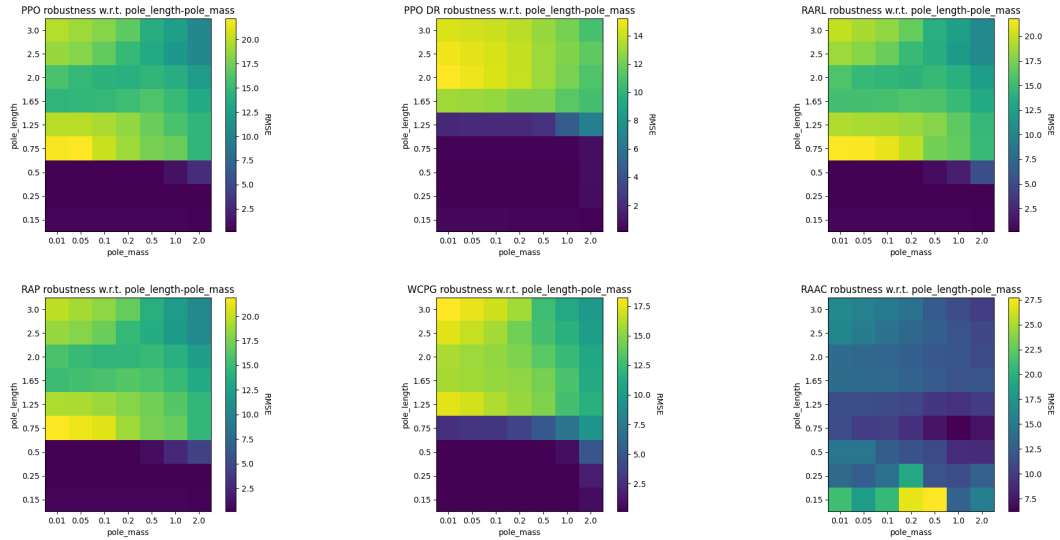


Figure 4.11: Robustness performance heatmaps with two varying disturbances (pole length and pole mass with default values 0.5 and 0.1) in cartpole for each baseline.

Next, Figure 4.12 shows the performance heatmaps with varying drone mass and action noise from normal distribution of varying scales in the quadrotor task. All

the baselines have similar heatmap patterns but differ in the RMSE scales. Specifically PPO, PPO DR, RARL, RAP and RAAC have roughly equal RMSE scales while WCPG has higher RMSE values on average compared to the others, indicating WCPG’s weaker ability to generalize facing the mixture of system parameter change and action noise in this harder task. Then again some common patterns are observed from the heatmaps: (i) smaller drone masses are harder for control than larger ones and incurs higher RMSE regardless of action noise as seen in the bottom region; (ii) related to point one for small drone mass, baselines perform better given larger action noise, which is especially obvious from the bottom rows of WCPG.

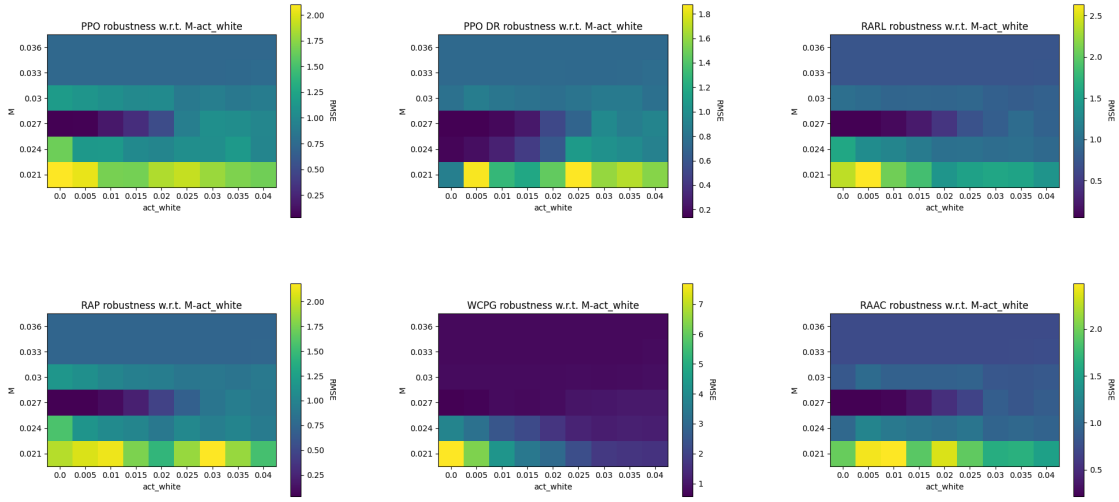


Figure 4.12: Robustness performance heatmaps with two varying disturbances (drone mass and action noise with default values 0.027 and 0) in quadrotor for each baseline.

In addition to the hybrid disturbance results presented in Figure 4.11 and 4.12, we also benchmark other combinations of the disturbances and include their performance heatmaps in Appendix B, providing a more complete analysis of the baselines’ robustness against different types of disturbance scenarios.

#### 4.2.4 Ablations

The above results section has demonstrated important patterns of the robust baselines regarding their training behaviors and generalization performance against various disturbances. To provide further insights for robustness benchmarking and guidance for designing future robust RL agents, we perform a series of ablations regarding the key parameters in the baselines, some of which are also brought up previously that

impact the training and robustness benchmark results.

### Domain randomization range

To start off, PPO DR has shown decent robustness against various disturbances in the cartpole task but the performance is not as well in the quadrotor task, and it is argued that the tuning of domain randomization parameters is the crucial factor. To demonstrate, we benchmark PPO DR with different levels of randomization ranges: low, mid and high as described below.

- **default:** pole length - 0.5, pole mass - 0.1, cart mass - 1.0
- **low:** pole length  $\sim \mathcal{U}(0.3, 0.7)$  pole mass  $\sim \mathcal{U}(0.05, 0.2)$  cart mass  $\sim \mathcal{U}(0.7, 1.3)$
- **mid:** pole length  $\sim \mathcal{U}(0.1, 1.0)$  pole mass  $\sim \mathcal{U}(0.01, 0.5)$  cart mass  $\sim \mathcal{U}(0.1, 2.0)$
- **high:** pole length  $\sim \mathcal{U}(0.1, 3.0)$  pole mass  $\sim \mathcal{U}(0.01, 2)$  cart mass  $\sim \mathcal{U}(0.1, 4.0)$

The default parameters are from the nominal cartpole environment and correspond to training a simple PPO agent without domain randomization. The mid-level parameters are used to train PPO DR and benchmark with other baselines in the results section. The low-level and high-level parameters are sampled from a smaller and a larger randomization range respectively.

| Tests           | PPO                | PPO DR (varying training DR range) |                   |                   |
|-----------------|--------------------|------------------------------------|-------------------|-------------------|
|                 |                    | low                                | mid               | high              |
| default         | $0.179 \pm 0.092$  | $0.148 \pm 0.074$                  | $0.16 \pm 0.087$  | $0.276 \pm 0.043$ |
| params x1.5     | $17.272 \pm 2.883$ | $11.379 \pm 3.166$                 | $0.325 \pm 0.07$  | $0.181 \pm 0.047$ |
| params x8       | $6.876 \pm 0.639$  | $6.133 \pm 0.787$                  | $5.058 \pm 1.089$ | $3.638 \pm 0.896$ |
| obs + act noise | $2.531 \pm 5.961$  | $0.501 \pm 0.108$                  | $0.629 \pm 0.1$   | $0.347 \pm 0.038$ |
| tap force       | $20.582 \pm 7.703$ | $9.987 \pm 6.934$                  | $2.349 \pm 1.954$ | $1.511 \pm 0.566$ |

Table 4.3: Domain randomization ablation in the cartpole task.

Table 4.3 shows the ablation results of varying level of domain randomization against different types of evaluation environments. Despite the default case or the nominal cartpole environment where PPO DR only has narrow improvement compared to PPO, it is observed that domain randomization can effectively counter the presence of disturbances which PPO falls short on. In the test cases where the system parameters are scaled by 1.5 times and 8 times, PPO DR has better performance when the randomization range also scale with the deviation of the test evaluation environment

to the nominal training environment. But PPO DR with high-level randomization turns out to be more effective than the one with mid-level randomization in the 1.5 times scaled cartpole, although the latter is trained on environments closer to the test environment. This might indicate a larger randomization range does not always sacrifice individual performance in exchange for better overall performance. A possible reason could be the larger randomization not only aids in robustness to disturbance but also induces better regularization for the RL training itself. Besides, domain randomization also generalizes to other forms of disturbances such as noise and external force. However, there is not a clear measure over the correlation between deviation of training environments to these test environments and the randomization range. From the current ablation, a larger randomization range always seems to be more versatile to different disturbances. Hence the tuning of domain randomization remains largely based on the availability of training resources, prior knowledge and extensive trial-and-error in these test cases.

### Adversary disturbance scale

A class of the benchmarked baselines are the adversarial RL methods which show limited robustness except to the disturbances that have been trained on from the results section. This naturally suggests combining multiple disturbances of interest during the adversarial training in order to achieve broader robustness. However, another important issue in adversarial RL is magnitude control over the adversary. It has been mentioned in previous work [51] that an overpowered adversary can easily undermine learning of the main policy. Yet a weak adversary is less effective in encouraging robust behaviors.

| Tests                               | PPO    | RARL (varying scale) |        |        |        | RAP (varying scale) |        |        |        |
|-------------------------------------|--------|----------------------|--------|--------|--------|---------------------|--------|--------|--------|
|                                     |        | 0.01                 | 0.1    | 0.5    | 1.0    | 0.01                | 0.1    | 0.5    | 1.0    |
| default                             | 0.215  | 0.526                | 4.744  | 2.978  | 6.845  | 0.225               | 0.211  | 0.244  | 0.226  |
| act noise $\sim \mathcal{N}(0, 1)$  | 0.197  | 0.247                | 3.587  | 2.107  | 7.092  | 0.226               | 0.211  | 0.478  | 0.231  |
| act noise $\sim \mathcal{N}(0, 3)$  | 0.264  | 0.337                | 5.027  | 2.279  | 6.463  | 0.798               | 0.256  | 1.582  | 0.269  |
| tap force $\sim \mathcal{U}(-1, 1)$ | 20.142 | 16.479               | 21.733 | 8.075  | 9.513  | 16.386              | 19.607 | 19.754 | 19.461 |
| tap force $\sim \mathcal{U}(-3, 3)$ | 25.542 | 27.377               | 26.929 | 11.461 | 12.738 | 25.134              | 26.398 | 28.189 | 27.024 |
| params x1.5                         | 17.617 | 17.861               | 18.018 | 5.787  | 6.331  | 16.870              | 16.932 | 17.555 | 17.415 |

Table 4.4: Adversary disturbance scale ablation in the cartpole task.

In Table 4.4 we demonstrate ablations of RARL and RAP with varying adversarial disturbance scales tested in differently disturbed environments. Naturally, RARL and

RAP perform better with external tap forces (row 4,5) and action noises (row 2,3) respectively, given they are trained against the corresponding type of disturbances as adversary. It is worth pointing out that the baseline PPO has similar scores as RAP in both the default nominal environment and disturbed environment with action noises. This may indicate PPO is already equipped with intrinsic robustness to action disturbances, such robustness is even competitive to RAP when the disturbance is small (row 2). On the other hand, although RARL displays poor performance in both the nominal and action disturbed environments, it generalizes better than the others when system parameters are changed.

### Risk measure threshold

Another class of the benchmarked robust baselines are the distributional RL methods. They are advantageous in that prior information of the disturbance is not needed for training, and there is flexibility in choosing a risk measure for policy optimization. In our benchmark, both WCPG and RAAC use  $\text{CVaR}_\alpha$  of the returns as the risk measure, and an important design decision is picking the risk measure threshold  $\alpha$ . Specifically, the threshold implicitly controls the weightings of learning data (e.g. lower performance data has higher focus) and affects how conservative each policy update is, which is eventually reflected as the final policy’s robustness.

| Tests           | PPO    | WCPG (varying $\alpha$ ) |        |       |        | RAAC (varying $\alpha$ ) |       |        |        |
|-----------------|--------|--------------------------|--------|-------|--------|--------------------------|-------|--------|--------|
|                 |        | 0.1                      | 0.3    | 0.5   | 1.0    | 0.1                      | 0.3   | 0.5    | 1.0    |
| default         | 0.202  | 0.585                    | 0.279  | 0.219 | 0.238  | 17.143                   | 6.188 | 12.352 | 21.721 |
| params x1.5     | 17.656 | 5.659                    | 10.305 | 6.703 | 6.918  | 11.957                   | 4.678 | 7.03   | 11.311 |
| obs + act noise | 2.135  | 0.649                    | 0.369  | 0.328 | 0.319  | 15.928                   | 5.956 | 11.645 | 22.757 |
| tap force       | 20.476 | 8.276                    | 15.608 | 10.34 | 10.157 | 18.704                   | 8.538 | 13.766 | 22.35  |

Table 4.5:  $\text{CVaR}_\alpha$  threshold ablation in the cartpole task.

Table 4.5 shows WCPG and RAAC’s performance given different risk measure thresholds for training and under various testing environments in cartpole. It should be noted that the  $\text{CVaR}$  threshold  $\alpha = 1$  is equivalent to the mean of the return distribution, so RAAC with  $\alpha = 1$  is similar to a nominal actor-critic method except with more complex parameterization and training for the value functions. For WCPG, it outperforms the other baselines under observation-action noise by a large margin, but only slightly win over RAAC under external force. Interestingly the WCPG variants show similar scores in the noisy environment to the default nominal environment, which might indicate small deviation between the two environments and hence fits

well with WCPG’s Gaussian return parameterization. However, it is also observed that there is not a single threshold that achieves the best performance on average over all the tested environments. For RAAC, it only has better performance with system parameter change but also not a clear margin compared to WCPG. Nonetheless, there seems to be a consistent trend that RAAC with threshold  $\alpha = 0.3$  is the best variant compared to other thresholds over the different disturbances. This observation is different to WCPG, and a possible reason is how the two baselines gets threshold differently in training. Concretely, instead of using a fix threshold as in RAAC, WCPG requires sampling a new threshold  $\alpha_i \sim \mathcal{U}(0.01, \alpha_{\max})$  for each episode (where  $i$  is current training episode index and  $\alpha_{\max}$  is the actual parameter we ablate over), which adds higher stochasticity and complicates the comparisons.

### 4.3 Transfer performance

There has been a rise in using simulated environments for learning-based control, especially in RL where the demand for data and training time is far greater than traditional methods. However, simulation can still fall short in producing a faithful representation of the real-world dynamics, motivating the need for incorporating real data in training with simulation. Transfer learning, domain adaptation and sim-to-real are all related topics trying to address this problem. Without proper adaptation or transfer, the trained controllers could neglect important yet hard-to-model correlations in the target environment or falsely overfit to nonidealities of the simulation. Therefore, the **safe-control-gym** benchmark also values the ability to adapt with extra data from a target domain, which is another crucial aspect of safety to deploy RL controllers. In this section, we benchmark several works on transfer learning in RL, with a focus on distribution matching methods; we compare their transfer performance when they are trained in a source domain with limited target data, and then evaluated in the target domain.

#### 4.3.1 Task Description

The benchmark tasks for transfer performance are based on the cartpole stabilization task. To perform domain transfer, the baselines require a source environment as the basis for learning and a target environment to query data for adaptation. In this section, the nominal cartpole system in **safe-control-gym** is used as the source environment and three different target environments (easy, mid, hard) are designed for

benchmarking as shown below. Notably, the easy and mid-level targets involve static change(s) of system parameters while the hard-level target involves state-dependent changes in dynamics. The three targets vary in similarity to the source environment and hence vary in difficulty for transfer learning.

- **source**: the base environment used by the transfer baselines for domain adaptation. It uses the default cartpole system parameters: pole length 0.5 m, pole mass 0.1 kg, and cart mass 1.0 kg.
- **target-easy**: a target environment to transfer/adapt to. It differs from the base environment in one system parameter: pole length 0.6 m.
- **target-mid**: a harder target environment that scales all three system parameters by 120%: pole length 0.6 m, pole mass 0.12 kg, and cart mass 1.2 kg.
- **target-hard**: another harder target that scales system parameters by 120% and further applies state-dependent damping: joint damping on the cart-pole joint with coefficient 0.06 and linear damping on the cart with coefficient 0.06.

Given a source and target environment pair, we consider the transfer task setting that consists of three components: (i) policy pretraining; (ii) target data collection and adaptation; and (iii) policy fine-tuning. Most of the transfer baselines considered can be covered by this setting. Specifically, we use only the source environment for policy pretraining, assuming no prior access to the target. Target adaptation and fine-tuning are often performed jointly, and we set a limited target data query of 500,000 control steps. All the evaluations are performed on the target environments. Other settings such as episode length, reward and action space are kept the same as tasks in Section 4.1 and 4.2. In addition, for baselines that require modifying the source environment during training such as SimGAN [38], we expose the pole mass and cart mass as the system’s parameter space that can be set dynamically at each control step.<sup>1</sup> With this system parameter space, the easy target can be fully adapted by successfully system identification, the mid-level target can only be partially adapted and the hard target requires dynamic adaptation given the state-dependent dampings. Consequently, all the considered baselines can be benchmarked on transfer performance given this curriculum of increasingly difficult target environments.

---

<sup>1</sup>Limited by PyBullet’s API, pole length is not included in the parameter space since it requires modifying and reloading the URDF file at each step, which could burden simulation performance.

### 4.3.2 Baselines

The following baselines are evaluated

- **PPO [57] and SAC [33]**. Similar to previous sections, they are general baselines for comparison. The nominal implementations do not consider transfer scenarios, so their transfer performance will rely only on the algorithms' intrinsic robustness.
- **fine-tuning (FT)**. Building on the nominal implementations, PPO and SAC are first pretrained in the simulation or source environment, then further fine-tuned given a limited amount of target data. This also serves as a general baseline with minimal effort in adaptation.
- **Domain Randomization (DR) [55, 64, 10]**. Similar to the baseline in Section 4.2 a RL agent is trained under environments with randomized parameters. So instead of fitting to one specific target environment, this approach potentially adapts to multiple targets. The transfer performance then relies on the robustness attained from the DR training.
- **SimGAN [38]**. In order to mimic the target environment with a source simulator, this approach uses RL to learn a set of simulator parameters that are optimized to match the distribution between target and source trajectories. A Generative Adversarial Network (GAN) [30] is trained with loss  $\mathcal{L}_{\text{disc}}$  to distinguish between source and target data. Target data is pre-collected before any training, and used in all subsequent GAN updates. The simulator agent is implemented as PPO and trained with reward  $r = \log(D_\phi(s, a, s')/(1 - D_\phi(s, a, s')))$ . Along with GAN and simulator agent training, another PPO control agent is trained using source data or data produced by the simulator only. Ideally as the discriminator and simulator agent improves, the gap between the target and simulated environment should narrow and the control agent should have better transfer performance.

$$\mathcal{L}_{\text{disc}} = -\mathbb{E}_{\tau_T}[\log D_\phi(s, a, s')] - \mathbb{E}_{\tau_S}[\log(1 - D_\phi(s, a, s'))]$$

$\tau_T$  — target trajectory distribution

$\tau_S$  — source trajectory distribution

$D_\phi(s, a, s')$  — discriminator output given transition tuple  $(s, a, s')$

- **GAT [35]**. This work uses regression to train a forward dynamics model given target data and an inverse dynamics model given source or simulator data. By chaining the two trained models, an action transformation can be derived that bridges between the target and source dynamics. Then the simulator and the action transformation together can act as an approximation to the target environment and be used in planning. In our implementation, we train a PPO agent  $\pi_\theta$  with the approximated target environment.

$$\begin{aligned} \text{target forward model : } s' &= f(s, a) \\ \text{simulator inverse model : } a &= f_{\text{sim}}^{-1}(s, s') \\ \text{action transformation : } \hat{a} &= f_{\text{sim}}^{-1}(s, f(s, a)), \quad a \sim \pi_\theta(\cdot|s) \end{aligned}$$

- **GARAT [19]**. Instead of using two separate dynamics models as in GAT, GARAT parameterizes the action transformation as one neural network  $\pi_g$ . It leverages GAN training similar to SimGAN for distribution matching between target and source trajectories, and uses reward  $r = -\log(1 - D_\phi(s, a, s'))$  to train the action transformation with RL. But GARAT differs from SimGAN in that new target data is collected as training proceeds, so the discriminator, action transformation and control agent are trained with on-policy target and source data. For implementation, both the action transformation and control agent are based on PPO.

$$\text{action transformation : } \hat{a} = \pi_g(s, a), \quad a \sim \pi_\theta(\cdot|s)$$

- **DARC [23]**. In DARC, distribution matching is implemented as auxiliary rewards  $\Delta r$ . The auxiliary reward is derived from minimizing the KL divergence between target and source trajectory distributions under the Maximum Entropy RL framework [33, 34], and simplifies to the difference of transition probabilities. But since transition dynamics is assumed to be unknown, two classifiers  $p(\cdot|s, a, s'), p(\cdot|s, a)$ , trained to distinguish between target and source transitions similar to the discriminator in SimGAN or GARAT, are used to substitute the transition probabilities via Bayes' rule. A SAC control agent is then trained

using the environment reward plus the auxiliary reward.

$$\begin{aligned}\Delta r(s, a, s') &= \log p_{\text{target}}(s'|s, a) - \log p_{\text{source}}(s'|s, a) \\ &= \log p(\text{target}|s, a, s') - \log p(\text{target}|s, a) \\ &\quad - \log p(\text{source}|s, a, s') + \log p(\text{source}|s, a)\end{aligned}$$

### 4.3.3 Results

The transfer baselines are evaluated under the three benchmark target tasks (easy, mid, hard) and we report the RMSE of adapted policy in the target environments. This section shows learning performances of the baselines when fine-tuning with target data, and compares their final performance after adaptation. Moreover, for baselines that rely on augmenting the source environment to approximate the target, we show the effectiveness of the augmented source to its target by examining the similarity of their trajectories both qualitatively and quantitatively. Overall, the benchmark results reveal how the baselines differ in the ability to explicitly learn and adapt given environment changes, which complements robustness and provides another axes of consideration to perform safe control with RL.

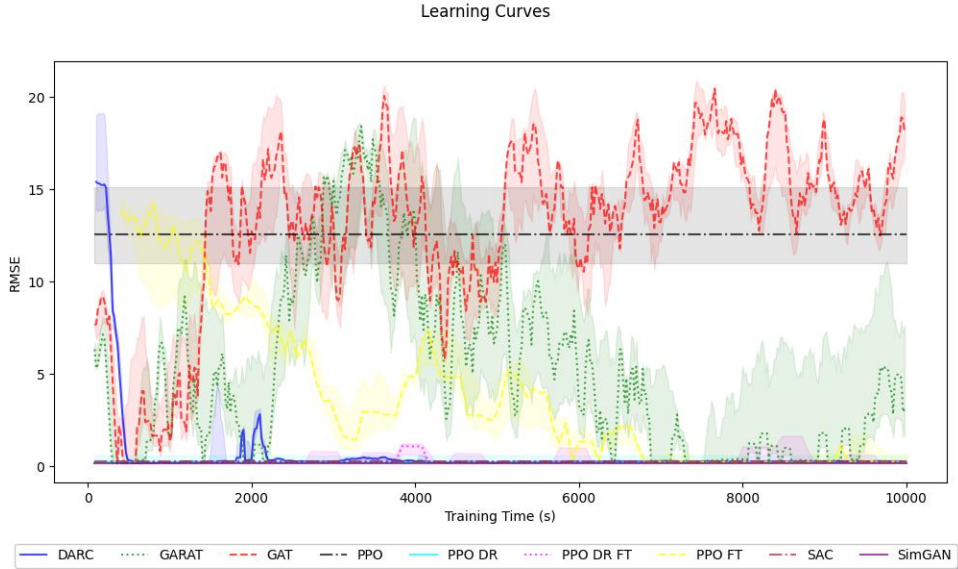


Figure 4.13: Transfer learning performance on cartpole target-easy.

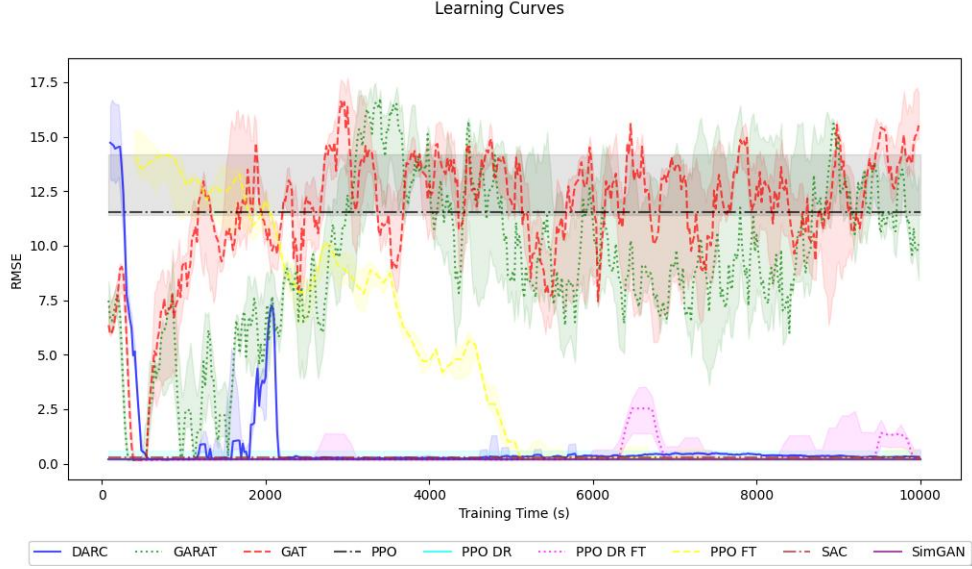


Figure 4.14: Transfer learning performance on cartpole target-mid.

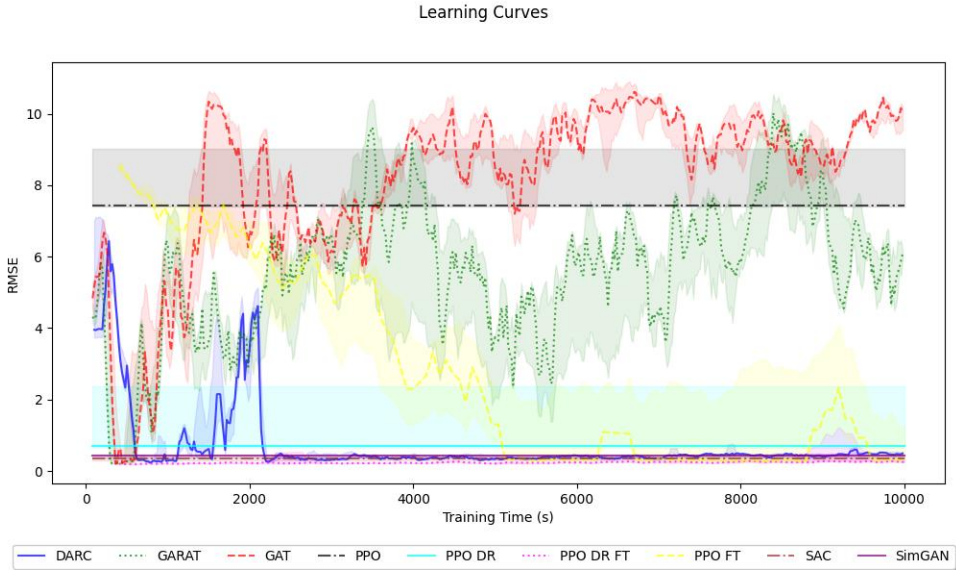


Figure 4.15: Transfer learning performance on cartpole target-hard.

Figure 4.13, 4.14, 4.15 demonstrate the learning performances of the transfer baselines under the three target environments. Specifically the x-axis denotes the amount of target data used in transfer with a maximal limit of 500,000 control steps or equivalently 10,000 seconds in target simulation time. The y-axis records the RMSE measured in the target environment using the current policy, but for baselines that do not leverage target data (PPO, PPO DR, SAC) or use target data separately from policy fine-tuning (SimGAN), we label them as flat learning curves with RMSE

evaluated using the final policy.

In all three targets, it is observed that PPO performs poorly while PPO FT starts off with poor performance but learns to adapt slowly. PPO DR has relatively low RMSE but not better than other baselines that explicitly leverage target information. In contrast, PPO DR with fine-tuning has one of the best transfer performances and even better than some dedicated transfer baselines. This set of comparisons highlights the importance of incorporating target data or more generally, reducing dissimilarity of the training data distribution to the target distribution, since domain randomization is also effective but may not even contain the target data. Interestingly, SAC shows decent results even if no target data is used, which means the source environment as a basis and the intrinsic robustness of MaxEnt RL training are sufficient to generalize to nontrivial environment changes. This again probes the question of how much robustness can be built into a safe controller and how can it be quantified. Despite the empirical benchmark results shown in Section 4.2, we also point out recent works that try to address such questions in theory [24, 25]. On the other hand, the superior performance of PPO DR FT compared to PPO FT in both learning speed and final target RMSE indicates the benefits of policy pretraining. Since a major bottleneck in transfer learning is collecting data from target environment and the pretrained model is derived purely in the source simulation, policy pretraining can be seen as an almost effortless enhancement to any transfer methods.

For the other transfer baselines, GAT and GARAT do not seem to perform well in general except GARAT in target-easy, and show heavy oscillations of transfer performances in evaluation. Such instability during learning can be due to the joint updates of the augmented source environment and the policy. The poor performance may also indicate grounding actions for transfer as proposed in GAT and GARAT are less effective given more complex environment changes. Despite this, it is observed that GARAT outperforms GAT in the three targets, showing distribution matching for transfer could be more effective or robust for RL-based controllers.

Lastly, SimGAN and DARC both have good transfer performances. SimGAN in particular outperforms most of the other baselines in all three targets, which can be the result of having privileged access to modify system parameters for augmenting the source environment. But noting that both target-mid and target-hard cannot be fully adapted given the restricted parameter space (refer to task description), this result still implies SimGAN’s ability for effective transfer. DARC also shows decent

RMSE performance but not as good as SimGAN or fine-tuning methods. A possible explanation is that it only relies on reward shaping for policy transfer instead of explicitly adapting the source environment to the target. Intuitively an augmented source can capture a wider class of dynamics, while reward shaping maintains the same dynamics as source but only shifts the data distribution. Hence, DARC might suffer when the source and target environments have clear differences in dynamics.

| Target Envs<br>Baselines | target - easy      | target - mid       | target - hard     |
|--------------------------|--------------------|--------------------|-------------------|
| PPO                      | $6.529 \pm 8.752$  | $9.398 \pm 8.497$  | $7.147 \pm 3.848$ |
| SAC                      | $0.197 \pm 0.095$  | $0.219 \pm 0.107$  | $0.351 \pm 0.113$ |
| PPO FT                   | $0.179 \pm 0.09$   | $0.171 \pm 0.086$  | $0.217 \pm 0.109$ |
| PPO DR                   | $0.475 \pm 1.104$  | $0.475 \pm 1.104$  | $0.398 \pm 0.487$ |
| PPO DR FT                | $0.176 \pm 0.089$  | $0.168 \pm 0.085$  | $0.195 \pm 0.098$ |
| SimGAN                   | $0.148 \pm 0.073$  | $0.153 \pm 0.076$  | $0.162 \pm 0.08$  |
| GAT                      | $18.217 \pm 6.322$ | $14.234 \pm 6.506$ | $9.728 \pm 2.117$ |
| GARAT                    | $2.475 \pm 3.23$   | $5.008 \pm 5.333$  | $3.747 \pm 4.673$ |
| DARC                     | $0.293 \pm 0.083$  | $0.298 \pm 0.145$  | $0.397 \pm 0.155$ |

Table 4.6: Final transfer performances (in RMSE) of baselines over the three target environments in cartpole. Highlighted cell shows the best result(s) in the column.

To summarize the benchmark results, Table 4.6 shows the final transfer performances of all the baselines on the three target tasks. It can be seen that SimGAN has the best overall transfer performances followed by the two fine-tuning methods, specifically PPO DR FT slightly outperforms PPO FT due to pretraining with domain randomization. SAC, PPO DR and DARC also exhibit some level of transfer but not as good as SimGAN or fine-tuning. The rest of the baselines seem to fail the transfer tasks, in particular, GAT and GARAT’s poor performances may be possibly due to insufficient hyperparameter tuning or incompatibility to the target environments.

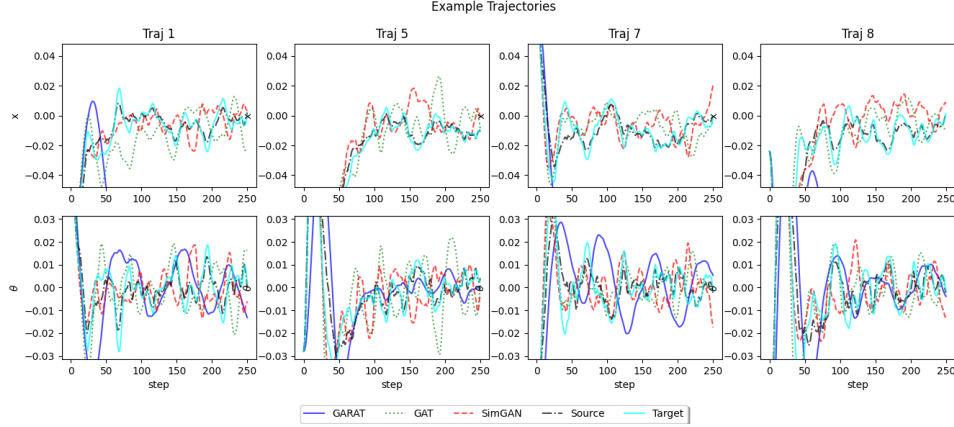


Figure 4.16: Example test trajectories with source, target and adapted environments using a policy trained in the target-mid environment.

Despite the difference in RMSE performance, the three baselines SimGAN, GAT and GARAT also vary in their augmented source environment. Since they leverage the augmented source for policy fine-tuning, a natural question that is key for domain transfer is how good the augmented environments can approximate the target. To answer this, we use a trained policy under the target-mid environment and sample trajectories with the augmented environments from the three baselines as in Figure 4.16. We add a trajectory generated from the unaugmented or source environment and another from the target environment for comparison, the initializations are kept the same for each environment. Similar comparison of trajectories for target-easy and target-hard can be found in Appendix C.

It is then observed that the trajectories of GAT and GARAT show larger dissimilarity to the target than either SimGAN or source environment. GARAT is also notably worse in adapting the cart position than the pole angle. This indicates GAT’s and GARAT’s augmented environments may not be effective for policy fine-tuning since they are not faithful or stable approximation to the desired target. Furthermore, it is noted that the trajectory from source environment is seemingly a better match to the target than trajectory from SimGAN’s augmented environment. This poses a question on what SimGAN learns in adapting the source to target that leads to good transfer performance, if its trajectory is not better in approximating the target.

| Baselines<br>Metrics | Target | Source | SimGAN | GAT   | GARAT |
|----------------------|--------|--------|--------|-------|-------|
| target - easy        |        |        |        |       |       |
| RMSE                 | 0.0    | 1.045  | 1.558  | 2.793 | 8.264 |
| MMD (gaussian)       | 0.014  | 0.056  | 0.038  | 0.087 | 0.592 |
| target - mid         |        |        |        |       |       |
| RMSE                 | 0.0    | 0.103  | 0.165  | 1.304 | 4.493 |
| MMD (gaussian)       | 0.002  | 0.006  | 0.008  | 0.063 | 0.675 |
| target - hard        |        |        |        |       |       |
| RMSE                 | 0.0    | 0.091  | 0.115  | 0.278 | 1.09  |
| MMD (gaussian)       | 0.002  | 0.02   | 0.008  | 0.028 | 0.458 |

Table 4.7: Similarity of trajectories between the target environment and source/adapted environments, using a policy trained in the target environment. Highlighted cell shows the best result in the row (excluding the first column). The first column is for self-comparison.

A possible explanation lies in the idea of distribution matching which is used by the three transfer baselines. In Table 4.7, we investigate this subject by computing the trajectory similarity between the target and the baselines with respect to two metrics: RMSE and Maximum Mean Discrepancy (MMD) [31]. For details about MMD, refer to Appendix D on the summary and analysis of using MMD as a similarity metric between environments based on trajectory samples.

For RMSE, it requires one-to-one comparison of two trajectories from the target and a baseline environment with the same initialization; the RMSE similarity scores also confirm the observation from Figure 4.16 that source environment does better in approximating the target than SimGAN. However, SimGAN shows better MMD scores in target-easy and target-hard, and is on par with the source environment in target-mid when taking into account the approximation error from the target’s self-comparison. To see why this result can justify SimGAN’s better transfer performance, we argue from two points:

1. **MMD is a more suitable trajectory similarity metric to evaluate the learned, augmented environments.** The MMD metric quantifies similarity between two probability distributions, but it can be approximately computed with only samples from the two distributions instead of requiring exact correspondence. This metric computation by sampling is also used for distribution matching when training the transfer baselines. Due to the affinity to the base-

line training, MMD should be a more proper metric to quantify performance of the augmented environments for the selected baselines.

## 2. MMD reveals trajectory similarity with respect to high-level features.

Distribution matching methods often focus on high-level features such as moments of the distributions. It means the augmented environments may not mimic the target in low-level details but the more abstract features that are deemed crucial by the method for transferring the policy, for example, correlations of states or behaviors at various time scales.

Together, these analyses should explain why SimGAN has good transfer performance despite not showing the closest match to the target trajectories, as supported from SimGAN’s better MMD scores rather than RMSE compared to the source environment.

### 4.3.4 Ablations

Similar to Section 4.1, 4.2, we then perform ablations for the transfer performance benchmark to investigate key design factors. Specifically we look at the influence of pretrained model/policy and target data, both of which have been brought up in the benchmark results section that contribute to better transfer performance. There are also other factors that are crucial for achieving domain transfer but are often tied to the specific baseline or environment, making it difficult to compare across different baselines, for example, the algorithm-specific discriminator tuning for distribution matching and the modifiable system parameters utilized by SimGAN. Therefore, we only benchmark on pretrained model and target data that are applicable to any transfer methods, and leave investigation of the algorithm-specific factors to future work that builds on top of the specific baseline.

#### Pretrained model

Policy pretraining is common in transfer learning to bootstrap initial performance and increase sampling efficiency. But not all pretrained models are equally beneficial for transfer, not to mention they also take on different engineering efforts to derive. In the **safe-control-gym** transfer benchmark, two important factors for pretrained models are the quality of the pretraining or source environment and the amount of pretraining budget. For the former, prior information from the target is often required to determine a good source environment, but the available prior is coupled with the

specific algorithm or the assumptions which makes it hard to ablate along. Here we only use the nominal cartpole or cartpole with domain randomization as the source, and focus on the impact of varying pretraining budget measured by the number of pretraining steps.

| Baselines<br>Pretrain Steps | PPO FT | PPO DR FT | SimGAN | GAT    | GARAT  | DARC  |
|-----------------------------|--------|-----------|--------|--------|--------|-------|
| 5,000                       | 0.182  | 0.168     | 21.283 | 13.852 | 10.921 | 0.253 |
| 20,000                      | 0.153  | 0.181     | 10.637 | 12.706 | 9.065  | 0.303 |
| 100,000                     | 0.176  | 0.437     | 7.891  | 9.614  | 9.672  | 0.226 |
| 500,000                     | 0.191  | 0.179     | 0.333  | 11.628 | 13.727 | 0.35  |

Table 4.8: Ablation on transfer baselines in target-mid environment by varying the amount of source data (in control steps) used for the pretrained policy. Highlighted cell shows the best result in the row.

As seen from Table 4.8, PPO FT, GAT and GARAT have a sweet spot in the number of pretraining steps that both leverages enough source data to bootstrap learning and also prevents overfitting to the source that hurts transfer performance. This is expected as the source environment is generally a coarse approximation of the target, and hence there is a limit on how much we can gain only from the source. PPO DR FT uses domain randomized source environment and requires more pretraining steps to start showing improvement on transfer performance. However, it manages to outperform other baselines in both the case of 5,000 and 500,000 pretraining steps. This can highlight the possible benefits of pretraining with domain randomization as better exploration for low-data regime and regularization to overfitting for high-data regime. In contrast, SimGAN has a even slower improvement to the pretraining budget, not until 500,000 steps does it starts showing similar transfer performance to the fine-tuning methods. Since SimGAN only uses the fixed pretrained policy during environment adaptation, it can lead to poorly adapted environment if the pretrained policy is insufficiently learned and hence affect the downstream policy fine-tuning. This becomes evident as SimGAN exhibits drastically bad performance with pretrained models in the low-data regime. Lastly, DARC’s transfer performances are slightly below the fine-tuning methods but do not show a clear pattern with the varying pretraining budget. This might be related to the different backbone agent used (SAC for DARC and PPO for the others). To be more specific, DARC uses the value functions from the pretrained model but it also learns augmented rewards for domain transfer that directly affects the value functions. These factors together might create mixing results that are not monotonic to the change of pretraining steps.

### Target data

The next important design factor for transfer learning is the amount of target data available for use. A key motivation of transfer learning in RL is to decrease the reliance on target data which is often the bottleneck for real-life applications. But in our transfer benchmark, all baselines are given a fixed budget of target data for fair comparison. Therefore, to complement the transfer benchmark, we also investigate how effective the baselines learn to transfer given varying amount of target data. We consider settings from low data (5,000 steps or 20 episodes) to high data (500,000 steps or 2000 episodes). The former can be related to few-shot transfer [58, 71] while the latter corresponds to our transfer benchmark setting. We do not consider the case of one-shot transfer (one episode) [69, 27] which is seemingly impractical for safe robotics.

| Target Steps \ Baselines |        |           |        |        |       |       |
|--------------------------|--------|-----------|--------|--------|-------|-------|
|                          | PPO FT | PPO DR FT | SimGAN | GAT    | GARAT | DARC  |
| 5,000                    | 13.956 | 0.136     | 0.118  | 6.413  | 0.203 | 1.216 |
| 20,000                   | 12.851 | 0.137     | 0.152  | 2.668  | 0.195 | 0.414 |
| 50,000                   | 8.751  | 0.139     | 0.152  | 13.358 | 9.81  | 0.362 |
| 200,000                  | 3.926  | 0.156     | 0.146  | 10.847 | 6.064 | 0.331 |
| 500,000                  | 1.037  | 0.164     | 0.156  | 11.102 | 5.008 | 0.298 |

Table 4.9: Ablation on transfer baselines in target-mid environment by varying the amount of target data (in control steps) used for fine-tuning/adaptation. Highlighted cell shows the best result in the row.

In Table 4.9, it is observed that PPO FT and DARC have obvious performance improvement with increasing target data, while PPO DR FT and SimGAN show the opposite trend. The former can be easily understood since PPO FT is simply training the policy in the target environment and with more target data it learns to correct suboptimal behaviors of the pretrained model due to source and target mismatch. Similarly, DARC trains its transfer policy with augmented rewards that are derived from comparing source and target dynamics, more target data naturally leads to more accurate reward signals and hence directly improves the transfer policy. It is also notable that DARC outperforms PPO FT in low-data regime and does not show a drastic improvement with more target data as PPO FT. This can be a possible result of DARC’s gradual adaptation of the policy and environment (captured by the augmented rewards) jointly, compared to PPO FT’s clear separation of policy pretraining in source and fine-tuning in target environment. As for PPO DR FT and SimGAN, they take over the best transfer performances over the different target step

settings, but surprisingly do not seem to improve given more target data. In particular, their transfer performance given only a few target data even surpasses other baselines’ performances in the high-data regime, indicating their superior transfer capability in both efficiency and effectiveness. Given PPO DR FT is pretrained with domain randomization, it is likely that the pretrained model has learned the general behavior to deal with varying cartpole dynamics and fine-tuning with too much target data muddles up such behavior instead. Such difference with PPO FT again emphasizes the impact of pretrained models, and shows us the necessity to consider specific traits of the method when balancing pretraining and fine-tuning. In contrast, SimGAN does not utilize domain randomization in pretraining, but still exhibits better performance with fewer target data. Since target data is generated via the pretrained model and only used for adapting the environment in SimGAN, we suspect too much target data might easily lead to overfitting to sub-optimal region of the state space in sacrifice of the more relevant ones for downstream fine-tuning. In this case, fewer data instead serves as an implicit regularization for environment adaptation which turns out better for transfer. Finally, GAT and GARAT also seem to perform better in the low-data regime and relatively worse with a nontrivial performance gap starting from the setting of 50,000 target steps. We similarly suspect the occurrence of overfitting at between 20,000 to 50,000 target steps that cuts off the improvement of transfer performance in these two baselines.

## 4.4 Summary

In this chapter, we select a set of representative RL approaches from the implemented baselines in **safe-control-gym** and benchmark their performances with respect to three dimensions of safety: constraint satisfaction, robustness to disturbances and transfer performance. For dealing with constraints, we show simple augmentations to constraint-agnostic RL baselines such as reward penalty and lagrangian method works decently after convergence, but do not prevent constraint violations during training. CMDP-based methods and safety critic methods can effectively reduce violations during learning, but could be sensitive to the constraint thresholds. For training robust agents, we show domain randomization gives relatively robust performance in various disturbance scenarios but at the cost of optimality. Other robust baselines based on adversarial training and distributional RL are good in certain scenarios but fall short in the others, showing limited generalizability. For transfer learning in RL, we show simple techniques of fine-tuning and domain randomization do allow transfer

to a new environment but are very limited. Distribution matching methods work better by incorporating knowledge from the target environment, but also heavily rely on the assumptions of applicable environments and parameter tuning; a method that performs well in the original work might not easily carry to another transfer scenario or robotic system. Overall, the chapter not only further demonstrates the safety features and various safe-learning baselines in **safe-control-gym**, but also showcases an extensive benchmark on the three crucial safety properties. The additional analysis of the benchmarked baselines gives insights on safe controller design and helps to guide future works of related topics.

# Chapter 5

## Conclusion

As more advanced learning-based control methods are developed and faster simulation technologies are popularized, control agents that utilize these two can now deal with increasingly difficult tasks and are showing promise to deploy directly in real-world systems. But before such promise can be delivered, safety remains to be a crucial consideration and it has been further complicated by the interplay with the new learning methods as well as training in simulation, especially for RL that are typically reliant on both.

This thesis takes on the challenge by proposing a benchmark for safe learning in robotics. To start off, we introduce the related literature in Section 2, focusing on benchmarking and safety-aware RL but as part of our bigger effort in the safe-learning survey [8]. Through the survey, we notice a current bottleneck in lacking sufficiently diverse, supportive safety benchmark for both control and RL approaches. To fill this vacancy we propose **safe-control-gym**. In Section 3 we cover the design of the **safe-control-gym** benchmark and demonstrate its usage with initial results, selected from our work [73]. With the benchmark, we compare the up-to-date safe-learning-based approaches with respect to three metrics of safety in Section 4. The benchmark results highlight important features and key design factors of these approaches, providing the basis for other research that utilizes these baselines in implementation or comparison.

Overall, through this line of works we have established the literature and software tools needed for advancing safe-learning control in robotics. The proposed benchmark also lies the foundation for continuous development and evaluation of other safe-learning-based methods in future research.

## Future directions

Considering the initiative of **safe-control-gym** as providing a diverse, supportive benchmark suite to develop and evaluate safe control approaches in robotics, a few future works are planned to expand the capabilities of the benchmark and to further close the gap from learning in simulation to real-life robot deployment.

Firstly, many new works on safe-learning control have come out since our work on the safe-learning survey and benchmark are publicated. Some of them showcase formulations that combine both the flexibility of RL and principled guarantees from traditional control, which **safe-control-gym** is lacking from the pool of baselines. Hence, an immediate direction is to continuously expand the safe control baselines, adding diversity to the benchmark and keeping track with recent progress.

Secondly, to provide better support for researchers to use **safe-control-gym** in their own use cases, it would be helpful to extend the benchmark with explicit interfaces to other common RL benchmarks such as StableBaselines3 or RLlib, as well as to other common environment suites and physics engines such as the recently open-sourced MuJoCo. Having wider compatibility to other well-established tools will also improve the usability of the benchmark for developing safe controllers.

In Section 4.3 we benchmark the transfer performance of several RL approaches, but the comparison is limited to transfer between different simulation environments. In practice it is more common to transfer from the simulation to the real world robot. Therefore, a desirable feature for future work is to implement a Sim2Real interface to a real robotic system in **safe-control-gym**, for example, a crazyflie setup that is mapped from the current quadrotor environment in Section 3.3.1.

Lastly, we also focus on bridging the gap between the RL and control communities in developing the survey and benchmark. Safety has been one important aspect that both RL and control approaches look into, and it has been the main topic of this thesis. Besides, another crucial aspect is the use of priors. Priors in RL often come in the form as past data or behavior policy, while priors in control often refers to nominal dynamic models; despite the difference they all have nonnegligible influence on the controller. Thus it will be another valuable direction if **safe-control-gym** can be extended to benchmark the impact of priors, specifically how prior trades off with learning data and how prior affects safety performance.

# Bibliography

- [1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. “Constrained policy optimization”. In: *International conference on machine learning*. PMLR. 2017, pp. 22–31.
- [2] Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.
- [3] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. “CasADi: a software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.
- [4] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* (1957), pp. 679–684.
- [5] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. “Safe model-based reinforcement learning with stability guarantees”. In: *Advances in neural information processing systems* 30 (2017).
- [6] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. “Conservative safety critics for exploration”. In: *arXiv preprint arXiv:2010.14497* (2020).
- [7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.
- [8] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. “Safe learning in robotics: From learning-based control to safe reinforcement learning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2021).
- [9] Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. “Measuring the reliability of reinforcement learning algorithms”. In: *arXiv preprint arXiv:1912.05663* (2019).

- [10] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Isaac, Nathan Ratliff, and Dieter Fox. “Closing the sim-to-real loop: Adapting simulation randomization with real world experience”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.
- [11] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. “Risk-constrained reinforcement learning with percentile risk criteria”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6070–6120.
- [12] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. “A lyapunov-based approach to safe reinforcement learning”. In: *Advances in neural information processing systems* 31 (2018).
- [13] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. “Lyapunov-based safe policy optimization for continuous control”. In: *arXiv preprint arXiv:1901.10031* (2019).
- [14] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models”. In: *Advances in neural information processing systems* 31 (2018).
- [15] The garage contributors. *Garage: A toolkit for reproducible reinforcement learning research*. <https://github.com/rlworkgroup/garage>. 2019.
- [16] Erwin Coumans and Yunfei Bai. “Pybullet, a python module for physics simulation for games, robotics and machine learning”. In: (2016).
- [17] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. “Safe exploration in continuous action spaces”. In: *arXiv preprint arXiv:1801.08757* (2018).
- [18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [19] Siddharth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, and Peter Stone. “An imitation from observation approach to transfer learning with dynamics mismatch”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3917–3929.
- [20] Yan Duan, Xi Chen, Rein Houthoofd, John Schulman, and Pieter Abbeel. “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 2016, pp. 1329–1338.

- [21] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. “An empirical investigation of the challenges of real-world reinforcement learning”. In: *arXiv preprint arXiv:2003.11881* (2020).
- [22] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. “Challenges of real-world reinforcement learning”. In: *arXiv preprint arXiv:1904.12901* (2019).
- [23] Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. “Off-dynamics reinforcement learning: Training for transfer with domain classifiers”. In: *arXiv preprint arXiv:2006.13916* (2020).
- [24] Benjamin Eysenbach and Sergey Levine. “If maxent rl is the answer, what is the question?” In: *arXiv preprint arXiv:1910.01913* (2019).
- [25] Benjamin Eysenbach and Sergey Levine. “Maximum entropy rl (provably) solves some robust rl problems”. In: *arXiv preprint arXiv:2103.06257* (2021).
- [26] Razvan V Florian. “Correct equations for the dynamics of the cart-pole system”. In: *Center for Cognitive and Neural Studies (Coneural), Romania* (2007).
- [27] Justin Fu, Sergey Levine, and Pieter Abbeel. “One-shot learning of manipulation skills with online dynamics adaptation and neural network priors”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4019–4026.
- [28] Javier Garcia and Fernando Fernandez. “A comprehensive survey on safe reinforcement learning”. In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.
- [29] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.
- [30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).
- [31] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Scholkopf, and Alexander Smola. “A kernel two-sample test”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773.

- [32] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gomez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. “Rl unplugged: A suite of benchmarks for offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7248–7259.
- [33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *arXiv preprint arXiv:1801.01290* (2018).
- [34] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. “Soft actor-critic algorithms and applications”. In: *arXiv preprint arXiv:1812.05905* (2018).
- [35] Josiah P Hanna and Peter Stone. “Grounded action transformation for robot learning in simulation”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.
- [36] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. “Deep reinforcement learning that matters”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [37] Lukas Hewing, Juraj Kabzan, and Melanie N Zeilinger. “Cautious model predictive control using gaussian process regression”. In: *IEEE Transactions on Control Systems Technology* 28.6 (2019), pp. 2736–2743.
- [38] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C Karen Liu, Sergey Levine, and Jie Tan. “Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2884–2890.
- [39] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. “URLB: Unsupervised reinforcement learning benchmark”. In: *arXiv preprint arXiv:2110.15191* (2021).
- [40] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. “AI safety gridworlds”. In: *arXiv preprint arXiv:1711.09883* (2017).

- [41] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. “RLlib: Abstractions for distributed reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3053–3062.
- [42] Qingkai Liang, Fanyu Que, and Eytan Modiano. “Accelerated primal-dual policy optimization for safe reinforcement learning”. In: *arXiv preprint arXiv:1802.06480* (2018).
- [43] Antonio Loquercio, Elia Kaufmann, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. “Deep drone racing: From simulation to reality with domain randomization”. In: *IEEE Transactions on Robotics* 36.1 (2019), pp. 1–14.
- [44] Carlos Luis and Jerome Le Ny. “Design of a trajectory tracking controller for a nanoquadcopter”. In: *arXiv preprint arXiv:1608.05786* (2016).
- [45] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.
- [46] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. “Active domain randomization”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 1162–1176.
- [47] Jean-Baptiste Mouret, Sylvain Koos, and Stephane Doncieux. “Crossing the reality gap: a short introduction to the transferability approach”. In: *arXiv preprint arXiv:1307.1870* (2013).
- [48] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. “Behaviour Suite for Reinforcement Learning”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=rygf-kSYwH>.
- [49] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. “Risk averse robust adversarial reinforcement learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8522–8528.

- [50] Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P Schoellig. “Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 7512–7519.
- [51] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. “Robust adversarial reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2817–2826.
- [52] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: <http://jmlr.org/papers/v22/20-1364.html>.
- [53] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. “Epopt: Learning robust neural network policies using model ensembles”. In: *arXiv preprint arXiv:1610.01283* (2016).
- [54] Alex Ray, Joshua Achiam, and Dario Amodei. “Benchmarking Safe Exploration in Deep Reinforcement Learning”. In: (2019).
- [55] Fereshteh Sadeghi and Sergey Levine. “Cad2rl: Real single-image flight without a single real image”. In: *arXiv preprint arXiv:1611.04201* (2016).
- [56] Harsh Satija, Philip Amortila, and Joelle Pineau. “Constrained markov decision processes via backward value functions”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8502–8511.
- [57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).
- [58] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical networks for few-shot learning”. In: *Advances in neural information processing systems* 30 (2017).
- [59] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. “Learning to be safe: Deep rl with a safety critic”. In: *arXiv preprint arXiv:2010.14603* (2020).
- [60] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. “Worst cases policy gradients”. In: *arXiv preprint arXiv:1911.03618* (2019).

- [61] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. “Learning for safety-critical control with control barrier functions”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 708–717.
- [62] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minh Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. “Recovery rl: Safe reinforcement learning with learned recovery zones”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922.
- [63] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. “Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3612–3619.
- [64] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.
- [65] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. “dm\_control: Software and tasks for continuous control”. In: *Software Impacts* 6 (2020), p. 100022.
- [66] Nuria Armengol Urpi, Sebastian Curi, and Andreas Krause. “Risk-averse offline reinforcement learning”. In: *arXiv preprint arXiv:2102.05371* (2021).
- [67] Joaquin Vanschoren, Serena Yeung, and Maria Xenochristou. *NeurIPS 2021 Datasets and Benchmarks Track*. 2021. URL: <https://neurips.cc/Conferences/2021/CallForDatasetsBenchmarks> (visited on 05/23/2022).
- [68] Eugene Vinitzky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. “Robust reinforcement learning using adversarial populations”. In: *arXiv preprint arXiv:2008.01825* (2020).
- [69] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems* 29 (2016).

- [70] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. “Benchmarking model-based reinforcement learning”. In: *arXiv preprint arXiv:1907.02057* (2019).
- [71] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. “Generalizing from a few examples: A survey on few-shot learning”. In: *ACM computing surveys (csur)* 53.3 (2020), pp. 1–34.
- [72] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. “Mutual alignment transfer learning”. In: *Conference on Robot Learning*. PMLR. 2017, pp. 281–290.
- [73] Zhaocong Yuan, Adam W Hall, Siqu Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P Schoellig. “safe-control-gym: a Unified Benchmark Suite for Safe Learning-based Control and Reinforcement Learning”. In: *arXiv preprint arXiv:2109.06325* (2021).
- [74] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. “Cautious adaptation for reinforcement learning in safety-critical settings”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11055–11065.

## 3D Quadrotor Dynamics

Section 3.3.1 describes the benchmark environments in **safe-control-gym**, including a quadrotor system with different degrees of freedom. Here we summarize the configuration and dynamics of the 3D quadrotor environment to complement the environment descriptions. Figure A.1 shows the crazyflie drone configuration<sup>1</sup> used in the quadrotor environment. The positive x axis corresponds to the front direction and the motor ordering follows the crazyflie URDF in **safe-control-gym**.

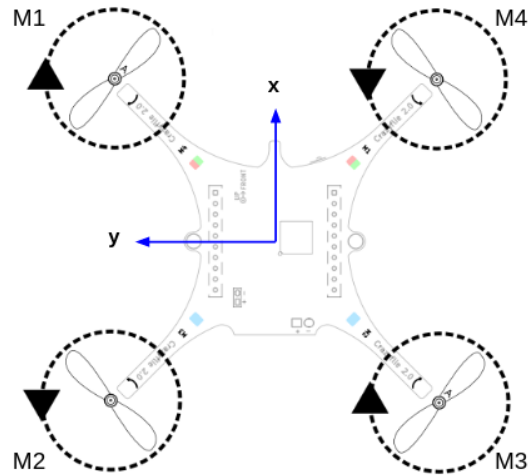


Figure A.1: Crazyflie diagram in the "X" configuration with motor directions and ordering used in **safe-control-gym**.

The full dynamics of 3D quadrotor is based on the shown configuration. The state consists of the drone positions, velocities, orientations in Euler angels with respect

<sup>1</sup>The configuration figure is adapted from Bitcraze tutorial's section "Unpacking the Crazyflie" (<https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/>).

to the inertial frame and the body rates with respect to the drone's body frame  $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]^T$ . The body rates can be converted to the angular velocities in inertial frame  $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$  but are more concise for dynamics equations. The following gives a summary of the 3D quadrotor dynamics based on the work [44].

Given the rotation matrix  $R_{ob}$  from the body frame to the inertial frame, equation A.1a describes the linear equation of motion.

$$\begin{aligned}
 R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \\
 R_z(\psi) &= \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_{ob} = R_z(\psi)R_y(\theta)R_x(\phi) \\
 \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} &= R_{ob} \begin{bmatrix} 0 \\ 0 \\ \frac{\sum_i T_i}{m} - g \end{bmatrix} \tag{A.1a}
 \end{aligned}$$

The angular dynamics is expressed in equations A.2a and A.2b, where  $M_b$  is the motor torque in body frame,  $l$  is the quadrotor arm length and  $I$  is the moment of inertia. The two parameters  $k_m$ ,  $k_f$  are the thrust and torque coefficients satisfying  $T_i = k_f \omega_i^2$ ,  $\tau_i = k_m \omega_i^2$ , where  $\omega_i, T_i, \tau_i$  are the rate of rotation, thrust and torque of the  $i$ -th motor respectively.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A.2a}$$

$$M_b = \begin{bmatrix} \frac{l}{\sqrt{2}}(T_1 + T_2 - T_3 - T_4) \\ \frac{l}{\sqrt{2}}(-T_1 + T_2 + T_3 - T_4) \\ \frac{k_m}{k_f}(T_1 - T_2 + T_3 - T_4) \end{bmatrix}, \quad \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \left( M_b - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{A.2b}$$

# Appendix B

## Robustness Heatmaps

Section 4.2 demonstrates the results of robustness benchmarking in **safe-control-gym**, including two performance heatmaps of the trained robust baselines against two varying types of disturbance. Despite the two, the following includes other performance heatmaps with different combinations of the disturbances on the cartpole stabilization and quadrotor tracking tasks.

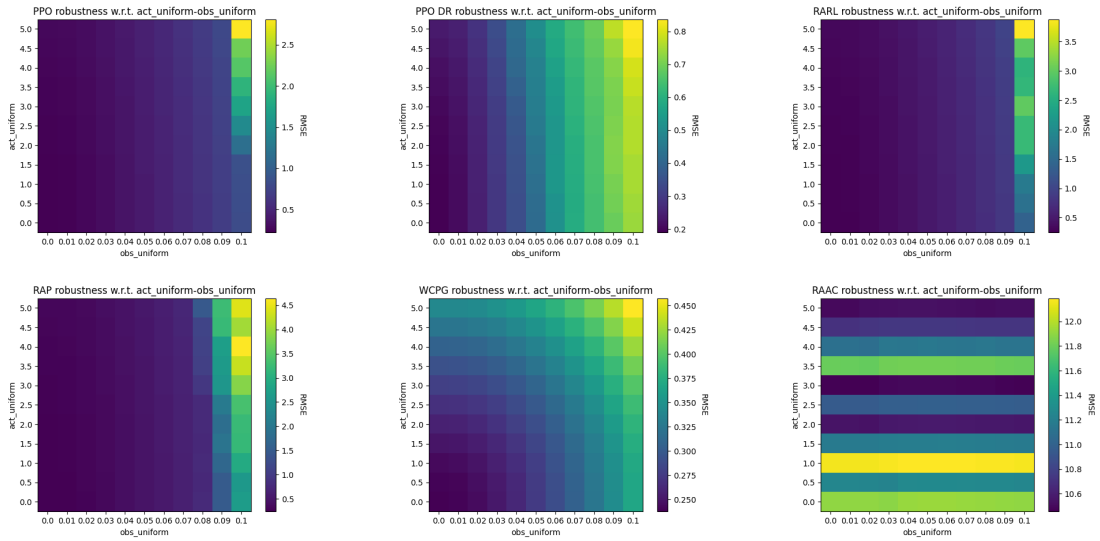


Figure B.1: Robustness performance heatmaps with varying action noise and observation noise in cartpole for each baseline.

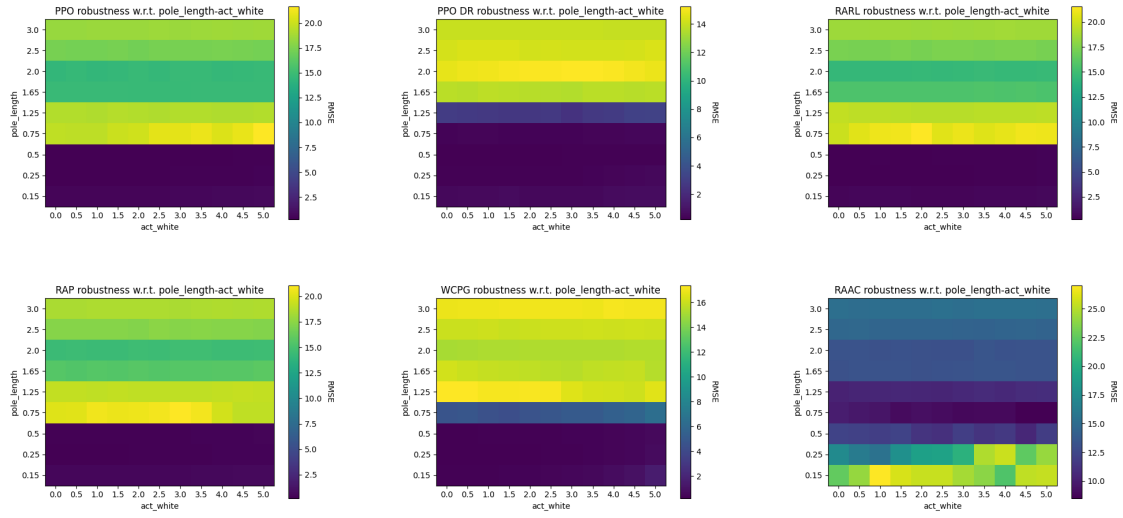


Figure B.2: Robustness performance heatmaps with varying pole length and action noise in cartpole for each baseline.

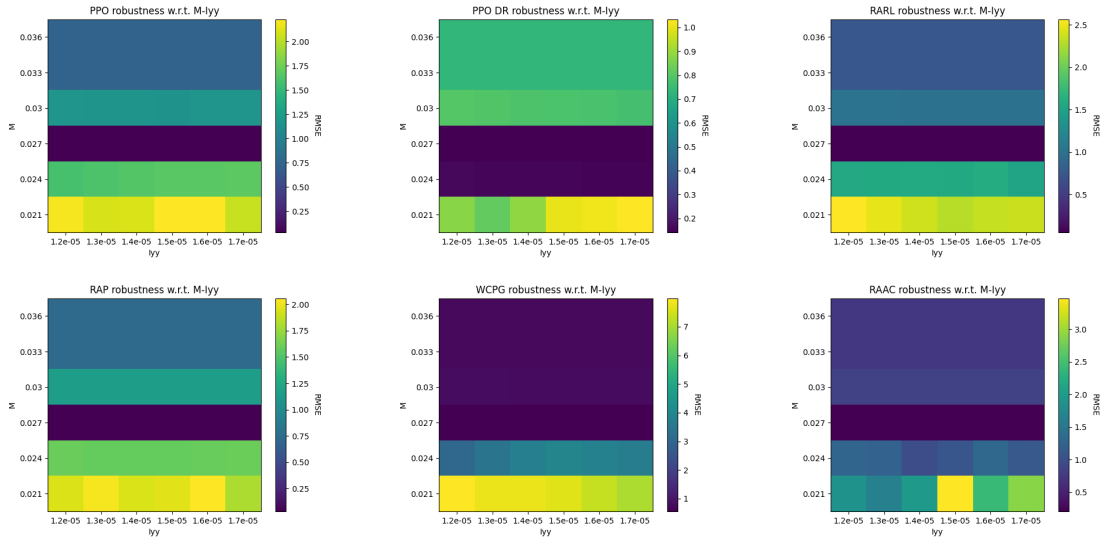


Figure B.3: Robustness performance heatmaps with varying drone mass and moment of inertia in quadrotor for each baseline.

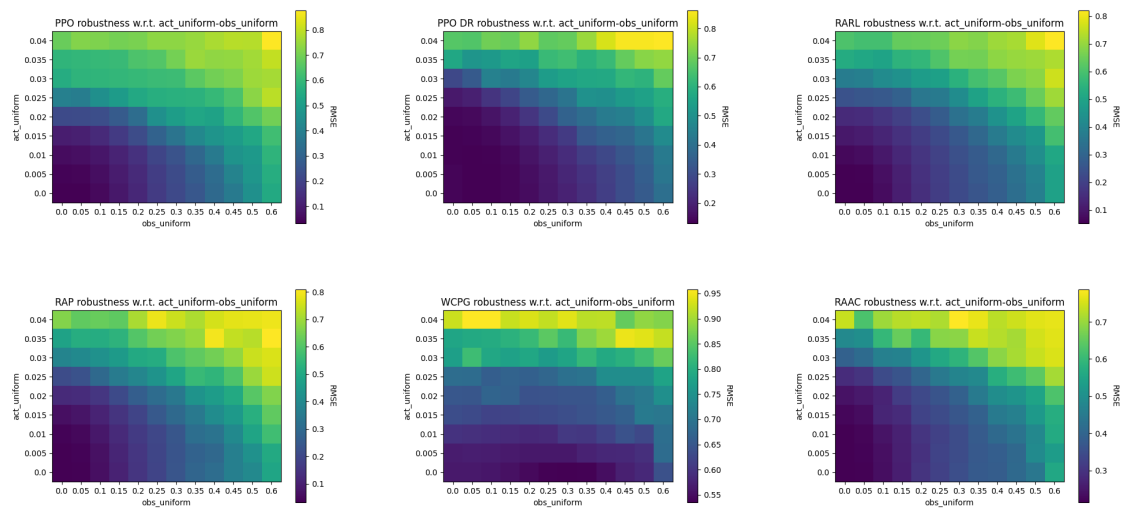


Figure B.4: Robustness performance heatmaps with varying action noise and observation noise in quadrotor for each baseline.

# Appendix C

## Augmented Environment Trajectories

As part of the benchmark results on transfer performance, Section 4.3 includes example trajectories of the baselines' augmented environments in target-mid cartpole. The following shows their other example trajectories in the target-easy and target-hard cartpole environments.

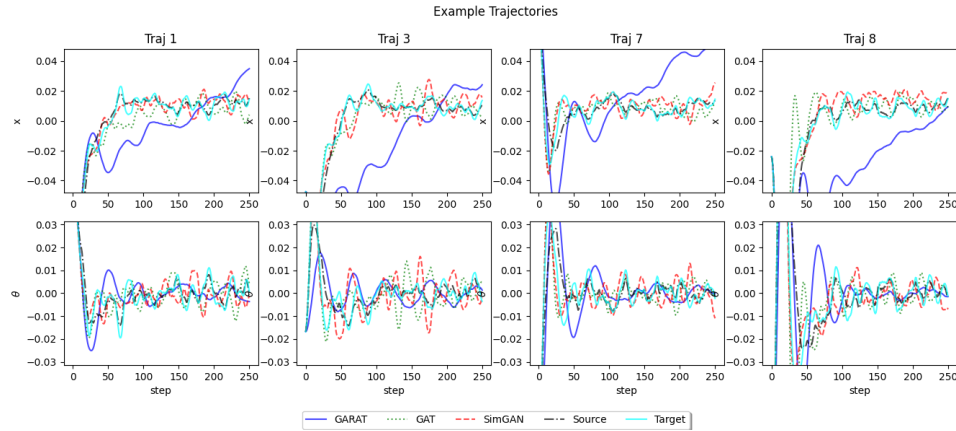


Figure C.1: Example test trajectories with source, target and adapted environments using a policy trained in the target-easy environment.

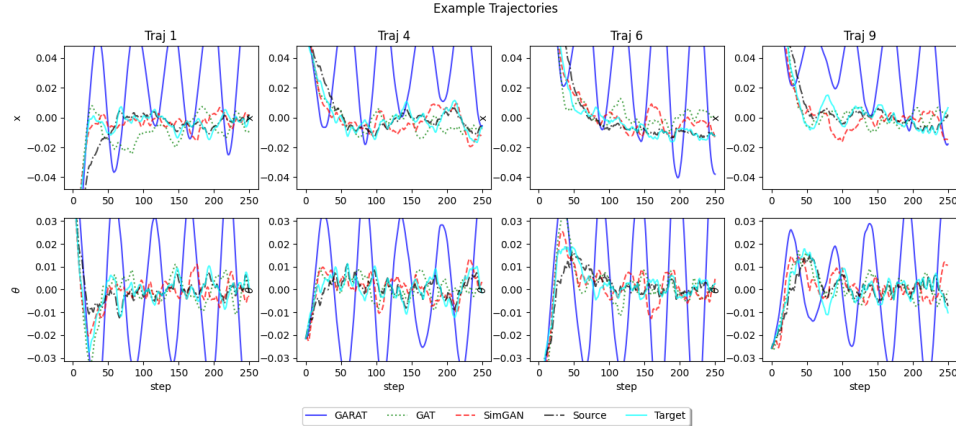


Figure C.2: Example test trajectories with source, target and adapted environments using a policy trained in the target-hard environment.

# Appendix D

## MMD Similarity Metric

Maximum Mean Discrepancy (MMD) is used in Section 4.3 to analyze the quality of learned augmented environments, specifically, it serves as a similarity metric between the augmented environments to the reference target environment. MMD is a valid distance metric for probability distributions and can be approximately computed via samples from the two compared distributions  $p, q$  directly as in Equation D.1a. The computation requires a continuous kernel function  $k(\cdot, \cdot)$  corresponding to some feature map on the distribution samples  $x \sim p, y \sim q$ , which is advantageous when the probability density functions are not given. For more detailed analysis on MMD, refer to the work [31].

$$\begin{aligned} \text{MMD}^2[p, q] &= \mathbf{E}_{x, x'}[k(x, x')] + \mathbf{E}_{y, y'}[k(y, y')] - 2\mathbf{E}_{x, y}[k(x, y)] \\ &\approx \frac{1}{m^2} \sum_i^m \sum_j^m k(x_i, x_j) + \frac{1}{n^2} \sum_i^n \sum_j^n k(y_i, y_j) - \frac{2}{mn} \sum_i^m \sum_j^n k(x_i, y_j) \end{aligned} \quad (\text{D.1a})$$

To demonstrate how MMD is effective in quantifying the similarity between two environments, we use the target-mid cartpole as the reference environment and calculate its MMD with various perturbed environments. We use a Gaussian kernel and treat each  $(s, a, s')$  transition tuple as a sample (by concatenation). The standard deviation  $\sigma$  serves as a tunable blur coefficient that controls the finest level of detail that should be handled by the metric and here we use  $\sigma = 10$  as the default.

$$k(x, y) = \exp(-\|x - y\|_2^2 / 2\sigma^2)$$

To derive the MMD of each augmented environment and the target environment, we

collect 5000 samples from each ( $m = n = 5000$ ) by running test trajectories with a PPO agent trained in the target environment. Figure D.1 shows the resulting MMD scores and it can be observed that the scores are relatively lower for environments that are less perturbed, which does indicate a measure of environment similarity.

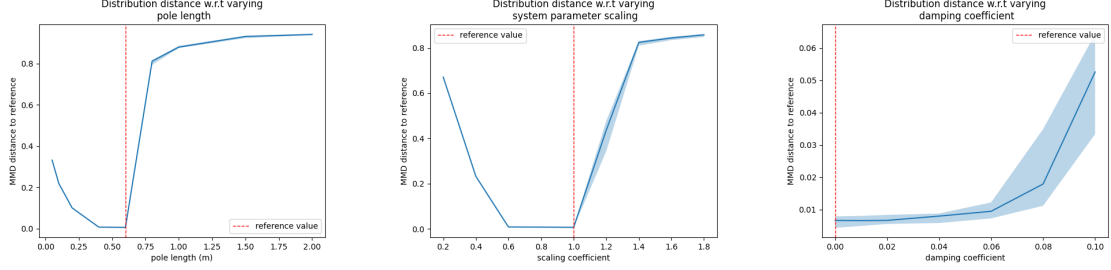


Figure D.1: MMD as a distribution distance to measure environment similarities. The three plots correspond to three kinds of perturbed environments based on target-mid cartpole: *Left*: only pole length changes. *Mid*: pole length, pole mass and cart mass are all scaled. *Right*: damping is added to the cart and pole joint. The vertical dash line represents the nonperturbed value in reference environment.